Linux From Scratch

Versão 6.1

Gerard Beekmans

Table of Contents

Prefácio

1. Introdução

Minhas aventuras em Linux começaram em 1998 quando eu fiz o download e instalei minha primeira distribuição. Após trabalhar com ela por certo tempo, descobri recursos que eu definitivamente gostaria de ver melhorados. Por exemplo, eu não gostei do arranjo dos scripts de inicialização ou da maneira que os aplicativos foram configurados pelo programa de instalação. Eu tentei várias distribuições alternativas, contudo cada uma tinha seus prós e seus contras. Finalmente, eu percebi que se eu quisesse ter plena satisfação com meu sistema Linux, eu teria que criar um desde o início, a partir do zero.

Como fazer isso? Eu resolvi não usar pacotes pré-compilados de qualquer tipo, nem discos de CD-ROMs ou discos de inicialização que instalassem utilitários básicos. Eu usaria meu sistema Linux atual para desenvolver meu próprio sistema persionalizado. Este sistema Linux "perfeito" teria então as qualidades de vários sistemas sem suas fraquezas associadas. No começo, a idéia pareceu desencorajadora, mas eu permaneci fiel ao propósito de que um sistema Linux poderia ser criado, a fim de atender às minhas necessidades e desejos, de uma forma muito melhor que um modelo padronizado que simplesmente não atendia ao que eu procurava.

Superando problemas tais como dependências recíprocas e erros em tempo de compilação, eu criei um sistema Linux inteiramente operacional e apropriado às minhas necessidades. Este processo permitiu também que eu criasse sistemas Linux compactos, personalizados, que são mais rápidos e que ocupam menos espaço em disco do que os sistemas operacionais tradicionais. Eu chamei este sistema de Linux From Scratch, ou LFS.

Enquanto eu compartilhava meus objetivos e experiências com outros membros da comunidade Linux, percebi que havia um grande interesse em minhas aventuras. Os sistemas LFS servem não somente para se adequar às especificações e exigências do usuário, mas servem também como uma oportunidade ideal de aprendizado para que programadores e administradores de sistema desenvolvam suas habilidades em Linux. O LFS foi desenvolvido também com esta segunda finalidade.

Este livro *Linux From Scratch* fornece as instruções necessárias para projetar e criar sistemas Linux feitos sob medida. Este roteiro enfoca o LFS e os benefícios de usar este sistema. Os usuários podem ditar todos os aspectos de seu sistema, incluindo a disposição dos diretórios, a instalação dos scripts e a segurança. O sistema resultante será compilado completamente a partir dos códigos fonte e o usuário poderá especificar onde, porque e como os programas serão instalados. Este livro permite que os leitores ajustem inteiramente o sistema Linux às suas próprias necessidades e permite aos usuários maior controle sobre seu sistema.

Eu espero que você tenha uma ótima experiência ao trabalhar em seu próprio sistema LFS e que aprecie os numerosos benefícios de ter um sistema que seja verdadeiramente *seu*.

--
Gerard Beekmans
gerard@linuxfromscratch.org

2. Público-alvo

Há várias razões para alguém querer ler este livro. A razão principal deve ser instalar um sistema Linux a partir dos códigos-fonte. Uma pergunta muito comum é “por que passar por todo este inconveniente de criar manualmente um sistema LFS quando você pode apenas fazer o download e instalar uma distribuição existente?” Esta é uma boa pergunta, e é a razão de ser desta seção do livro.

Uma razão muito importante para a existência do LFS é ajudar no aprendizado sobre como um sistema Linux trabalha internamente. Construir um sistema LFS demonstra como funciona o Linux, como as suas partes trabalham junto e como dependem entre si. Uma das melhores coisas que esta experiência de aprendizado fornece é a habilidade de personalizar o Linux ao seu próprio gosto e necessidade.

O maior benefício da implementação do LFS é permitir que os usuários tenham mais controle sobre o sistema sem ter que confiar na implementação do Linux de outra pessoa. Com o LFS, *você* está na cadeira do motorista e define cada aspecto do sistema, tal como a disposição da árvore de diretórios e as rotinas de inicialização. Você determina também onde, porque e como os programas serão instalados.

Outro benefício do LFS é poder criar um sistema Linux muito compacto. Ao instalar uma distribuição regular, você freqüentemente é forçado a incluir diversos programas que provavelmente não serão usados nunca. Estes programas desperdiçam espaço em disco, ou pior, recursos do processador central. Não é difícil criar um sistema LFS com menos de 100 megabytes (MB), o que é substancialmente menor do que a maioria de distribuições existentes. Isto ainda lhe soa como muito do espaço? Alguns de nós têm trabalhado para criar um sistema LFS muito pequeno. Nós construímos com sucesso um sistema especializado para funcionar com o Apache Web Server com aproximadamente 8MB de espaço em disco usado. Dispensar alguns adicionais poderia reduzir este espaço para 5 MB. Tente isso com uma distribuição regular! Este é somente um dos muitos benefícios de projetar sua própria versão do Linux.

Nós poderíamos comparar distribuições do Linux a um hamburger comprado em um fast-food — não temos nenhuma idéia do que se está comendo. O LFS, por outro lado, não lhe dá um hamburger, mas sim a receita para fazer o hamburger que você deseja. Isto permite aos usuários examinar a receita, suprimir ingredientes não desejados e adicionar seus próprios ingredientes para realçar o sabor do hamburguer. Quando você estiver satisfeito com sua receita, comece a prepará-la. E pode fazê-lo da forma que você preferir — grelhado, assado ou frito.

Uma outra analogia que nós podemos usar é comparar o LFS com uma casa terminada. O LFS fornece a planta estrutural de uma casa, mas é você quem vai construí-la. E o LFS mantém sua liberdade para ajustar a planta durante todo o processo de construção, personalizada segundo as suas necessidades e preferências.

Uma vantagem adicional de um sistema Linux personalizado é a segurança. Compilando o sistema inteiro a partir dos códigos-fonte, você pode examinar tudo e aplicar todos as coreções (patches) de segurança que quiser. Não é necessário esperar que alguém compile os binários com as correções que reparam alguma falha na segurança. Além disto, a menos que você examine a correção e a aplique você mesmo, você não terá nenhuma garantia de que o problema com o pacote binário foi corrigido adequadamente.

O objetivo do LFS é montar um sistema completo e funcional de nível elementar. Os leitores que não desejam criar seu próprio sistema LFS não vão se beneficiar das informações deste livro. Se você quiser somente saber o que acontece quando o computador inicializa, nós recomendamos o HOWTO “From Power Up To Bash Prompt” localizado em *http://axiom.anu.edu.au/~okeefe/p2b/* ou o website The Linux Documentation Project's (TLDP) em *http://www.tldp.org/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html*. Este HOWTO monta um sistema similar àquele que montamos neste lívro, mas se consentra estritamente em criar um sistema capaz de carregar um alerta de comando (prompt) do BASH. Considere seu objetivo. Se você deseja criar um

sistema Linux e aprender ao longo do processo, então este livro é sua melhor escolha.

Existem tantas boas razões para você criar seu próprio sistema LFS que não dá para listá-las todas aqui. Esta seção é somente a ponta do iceberg. Ao continuar em sua experiência com o LFS você sentirá o poder que a informação e o conhecimento verdadeiramente pode trazer.

3. Pré-requisitos

Este livro supõe que o leitor tenha um conhecimento razoável sobre o uso e a instalação de software em um sistema Linux. Antes de criar um sistema LFS, nós recomendamos a leitura dos seguintes HOWTOs:

- Software-Building-HOWTO
  http://www.tldp.org/HOWTO/Software-Building-HOWTO.html

  Este é um guia abrangente sobre a configuração e instalação de software UNIX “genérico” no Linux.
- The Linux Users' Guide
  http://www.linuxhq.com/guides/LUG/guide.html

  Este guia trata do uso de vários softwares para Linux.
- The Essential Pre-Reading Hint
  http://www.linuxfromscratch.org/hints/downloads/files/essential_prereading.txt

  Esta é uma sugestão da LFS escrita especificamente para os usuários novatos em Linux. Inclui uma lista de endreços com excelentes fontes de informação sobre muitos tópicos. Qualquer um que tente instalar um sistema LFS deve ter uma compreensão de muitos dos tópicos indicados nesta sugestão.

4. Requisitos do sistema anfitrião

O sistema anfitrião deve estar executando pelo menos um kernel 2.6.2 compilado com o GCC 3.0 ou superior. Há duas razões principais para esta exigência. Primeiro, o conjunto de testes da biblioteca Native POSIX Threading Library (NPTL) falha se o kernel do anfitrião não for compilado com o GCC 3.0 ou posterior. Em segundo, a versão 2.6.2, ou superior, do kernel é exigida para o uso do Udev. O Udev cria dispositivos dinamicamente, lendo diretamente do sistema de arquivos `sysfs`. Entretanto, o suporte para este sistema de arquivos somente foi implementado recentemente na maioria dos gerenciadores de dispositivos do kernel. Nós devemos garantir que todos os dispositivos críticos do sistema sejam criados corretamente.

Para determinar se o kernel do sistema anfitrião se adequa às exigências acima, execute o seguinte comando:

```
cat /proc/version
```

O comando deve produzir uma saída similar a isto:

```
Linux version 2.6.2 (user@host) (gcc version 3.4.0) #1
    Tue Apr 20 21:22:18 GMT 2004
```

Se os resultados do comando acima não indicarem que o kernel do sistema é 2.6.2 (ou posterior), ou que não foi compilado usando o GCC 3.0 (ou posterior), um sistema com estas especificações precisará ser instalado. Há duas formas de você resolver isto. Primeiro, veja se o fornecedor de seu sistema Linux disponibiliza uma atualização para o kernel 2.6.2. Se não oferecer, ou se você preferir não a instalar, então você mesmo pode compilar um kernel 2.6. As instruções para compilar o kernel e configurar o boot loader (supondo que você utilize o GRUB no sistema anfitrião) podem ser consultadas no Chapter 8. Esta segunda opção pode também ser vista como uma medida de suas habilidades atuais em Linux. Se esta segunda exigência for difícil demais, então este livro provavelmente não será muito útil para você no momento.

5. Convenções utilizadas neste livro

Para tornar as coisas mais fáceis de acompanhar, foram usadas algumas convenções tipográficas durante todo este livro. Esta seção contém alguns exemplos do formato tipográfico utilizado.

```
./configure --prefix=/usr
```

Este formato de texto é para ser digitado exatamente como visto, a menos que esteja destacado de outra maneira no texto explicativo. É também utilizada nas seções de explicação para identificar qual dos comandos está sendo descrito.

```
install-info: unknown option '--dir-file=/mnt/lfs/usr/info/dir'
```

Este formato (texto de largura fixa) mostra a saída de tela, provavelmente como o resultado de algum comando. Também é usado para mostrar os nomes de arquivo, como `/etc/ld.so.conf`.

Ênfase

Este formato é usado para diversas finalidades no livro, normalmente para enfatizar pontos ou artigos importantes.

http://www.linuxfromscratch.org/

Esta forma é utilizada para endereços eletrônicos, tanto para os de outros capítulos, ou seções, dentro do próprio livro como para HOWTOs, arquivos disponíveis para download, sites etc.

```
cat > $LFS/etc/group << "EOF"
root:x:0:
bin:x:1:
......
EOF
```

Este formato é usado principalmente para criar arquivos de configuração. O primeiro comando diz ao sistema para criar o arquivo `$LFS/etc/group` contendo tudo o que é digitado nas linhas seguintes, até a seqüência EOF (fimde arquivo, do inglês end of file) ser encontrada. Portanto, toda esta seção é geralmente digitada exatamente como aparece.

`[TEXTO A SER SUBSTITUÍDO]`

Este formato é usado para demarcar o texto que não deve ser datilografado como visto nem copiado-e-colado. O texto próximo explica os parâmetros ou as opções que devem ser utilizadas no lugar no texto demarcado.

`passwd(5)`

Este formato é usado para consultar a uma página específica do manual (doravante referida simplesmente como página “man”). O número dentro dos parênteses indica uma seção específica dentro da página **man**. Por exemplo **passwd** tem duas páginas man. Pelas instruções de instalação do LFS, aquelas duas páginas man serão encontradas em `/usr/share/man/man1/passwd.1` e `/usr/share/man/man5/passwd.5`. As duas páginas man têm informações diferente nelas. Quando o livro utilizar `passwd(5)` ele está se referindo especificamente a `/usr/share/man/man5/passwd.5`. O comando **man passwd** vai mostra a primeira página man que encontrar e que corresponda a “passwd”, que será a `/usr/share/man/man1/passwd.1`. Para este exemplo, você precisa executar **man 5 passwd** a fim de ler a página específica a qual o texto faz referência. É preciso mencionar que a maioria das páginas do man não têm nomes duplicados. Portanto, **man**

`[nome do programa]` geralmente é suficiente.

6. Estrutura

Este livro é dividido nas seguintes partes.

6.1. Part I - Introdução

A parte I explica algumas coisas importantes para iniciar a instalação do LFS. Esta seção fornece também informações gerais sobre o livro.

6.2. Part II - Preparando a configuração

A parte II descreve como preparar-se para o processo de construção do LFS particionando o disco, fazendo o download dos pacotes e compilando as ferramentas provisórias.

6.3. Part III - Construindo o sistema LFS

A parte III guia o leitor através da construção do sistema LFS — compilando e instalando todos os pacotes um por um, ajustando as rotinas de inicialização (boot scripts) e instalando o kernel. O sistema Linux resultante é o alicerce básico, onde os outros softwares podem ser instalados, para expandir o sistema como desejado. No final de cada seção há uma referência aos programas, bibliotecas e arquivos importantes que foram instalados.

7. Errata

O software utilizado para criar o sistema LFS está sendo constantemente atualizado e desenvolvido. Avisos de segurança e reparos de erros podem se tornar disponíveis depois de liberado este livro. Para verificar se há novas versões dos pacotes utilizados ou instruções desta versão do LFS que precisem de alguma modificação para remover vulnerabilidades de segurança ou para corrigir outros erros, por favor, acesse *http://www.linuxfromscratch.org/lfs/errata/6.1/* antes de proseguir com a configuração. Você deve anotar todas as mudanças encontradas lá e aplicá-las às respectivas seções do livro enquanto configura o seu sistema LFS.

Part I. Introdução

Chapter 1. Introdução

1.1. Como construir um sistema LFS

O sistema LFS será construído usando uma distribuição previamente instalada do Linux (tal como Debian, Mandrake, o Red Hat, ou o SuSE). Este sistema (o anfitrião) será usado como ponto de partida e fornecerá os programas necessários, incluindo um compilador, um editor de vínculos (linker), e um shell, para montar o novo sistema. Escolha a opção “desenvolvimento”, ou similar, durante a instalação da distribuição anfitriã para ter acesso a estas ferramentas.

Como alternativa à instalação de uma distribuição completa em sua máquina, você pode preferir usar o Linux From Scratch LiveCD. O LiveCD trabalha bem como um sistema de anfitrião, fornecendo todas as ferramentas que você necessita para acompanhar com sucesso as instruções deste livro. Adicionalmente, contém todos os pacotes de fontes, patches e uma cópia deste livro [n.t. do original, em inglês]. Assim quando você tem o LiveCD, nem conexão de rede nem downloads adicionais serão necessários. Para mais informação sobre o LFS LiveCD ou para fazer o download de uma cópia, visite *http://www.linuxfromscratch.org/livecd/*.

O Chapter 2 deste livro descreve como criar uma nova partição Linux e um sistema de arquivos nativo, onde o novo sistema LFS será compilado e instalado. O Chapter 3 explica que pacotes e patches são necessários fazer o download para construir um sistema LFS e como os armazenar no novo sistema de arquivos. O Chapter 4 discute a instalação e configuração de um ambiente de trabalho próprio ao processo de montagem do sistema LFS. Leia por favor o Chapter 4 com muito cuidado, pois ele tem explicações muito importantes que o leitor deve estar ciente antes do começar trabalhar com o Chapter 5 e seguintes.

O Chapter 5 explica a instalação de um certo número de pacotes que dão forma ao conjunto básico de desenvolvimento (ou "toolchain") que será usado para construir o sistema real no Chapter 6. Alguns destes pacotes são necessários para resolver dependências cruzadas ou circulares—por exemplo para compilar um compilador, é necessário ter um compilador.

O Chapter 5 mostra também ao usuário como construir uma primeira versão do jogo de ferramentas (que o LFS-book chama detoolchain), incluindo o Binutils e o GCC (por primeira versão queremos dizer basicamente que estes pacotes serão reinstalados uma segunda vez). A etapa seguinte é configurar a Glibc, a biblioteca C. A Glibc será compilada pelos programas do toolchain construídos em primeira versão. Então, uma segunda versão do conjunto de ferramentas será configurada. Desta vez, as ferramentas serão vinculadas dinamicamente ao Glibc recém-configurado. Os pacotes restantes do Chapter 5 são configurados usando este segundo conjunto de ferramentas. Quando isto é feito, o processo da instalação do LFS não mais depende da distribuição anfitriã, com exceção do kernel.

Uma explicação técnica detalhada sobre quando é possível isolar o novo sistema da distribuição anfitriã é dada no começo do Chapter 5.

No Chapter 6, o sistema LFS completo é construído. O programa **chroot** (change root) é usado para entrar em um ambiente virtual e inicializar um novo shell cujo o diretório de raiz seja definido na partição do LFS. Isto é muito similar a reinicializar e a instruir o kernel para montar a partição LFS como a partição root. O sistema não reinicializa realmente, mas faz um **chroot**porque criar um sistema inicializável requer o trabalho adicional que não é necessário neste momento. A vantagem principal de “chrooting” é permitir o uso do sistema anfitrião enquanto o LFS estiver sendo configurado. Enquanto espera a compilação de algum pacote terminar, o usuário pode abrir um console virtual diferente (VC) ou o desktop X e continuar usando seu computador normalmente.

Para terminar a instalação, o LFS-Bootscripts é configurado no Chapter 7, e o kernel e o boot loader (carregador

de sistemas) são configurados no Chapter 8. O Chapter 9 contém informações sobre como prosseguir na experiência do LFS para além deste livro. Depois que as etapas deste livro forem executadas, o computador estará pronto para reiniciar no novo sistema LFS.

Este é o processo resumido. As informações detalhadas de cada etapa são discutidas nos capítulos seguintes e nas descrições dos pacotes. As passagens que podem parecer complicadas serão esclarecidas, e tudo estará em seu devido lugar assim que o leitor embarcar na aventura LFS.

1.2. Changelog

Esta é a versão 6.1 do livro Linux From Scratch, datada de 9 de Julho de 2005. Se este roteiro tiver mais de seis meses, uma versão mais nova e melhor provavelmente já estará disponível. Verifique por favor um dos mirrors através da página *http://www.linuxfromscratch.org/*.

Abaixo está uma lista das mudanças feitas desde a sua liberação. Primeiro um sumário, depois um registro detalhado.

- Atualizado para:
  - Automake 1.9.5
  - Binutils 2.15.94.0.2.2
  - Bison 2.0
  - Bzip2 1.0.3
  - E2fsprogs 1.37
  - Expect 5.43.0
  - File 4.13
  - Findutils 4.2.23
  - GCC 3.4.3
  - Gettext 0.14.2
  - Glibc 2.3.4
  - Grep 2.5.1a
  - Grub 0.96
  - Iana-Etc 1.04
  - Iproute2 2.6.11-050330
  - LFS-Bootscripts 3.2.1
  - Libtool 1.5.14
  - Linux 2.6.11.12
  - Linux-libc-headers 2.6.11.2
  - M4 1.4.3
  - Man 1.5p
  - Man-pages 2.01
  - Module-init-tools 3.1
  - Perl 5.8.6

- Procps 3.2.5
    - Psmisc 21.6
    - Sed 4.1.4
    - Shadow 4.0.9
    - Sysvinit 2.86
    - Tar 1.15.1
    - Texinfo 4.8
    - Tcl 8.4.9
    - Udev 056
    - Util-linux 2.12q
    - Zlib 1.2.2

- Adicionado:

    - bash-3.0-fixes-3.patch
    - bash-3.0-avoid_WCONTINUED-1.patch
    - flex-2.5.31-debian_fixes-3.patch
    - glibc-2.3.4-fix_test-1.patch
    - gzip-1.3.5-security_fixes-1.patch
    - Hotplug 2004_09_23
    - mktemp-1.5-add_tempfile-2.patch
    - sysklogd-1.4.1-fixes-1.patch
    - tar-1.15.1-sparse_fix-1.patch
    - util-linux-2.12p-cramfs-1.patch
    - vim-6.0-security_fix-1.patch
    - zlib-1.2.2-security_fix-1.patch;

- Removido:

    - bash-3.0-display_wrap-1.patch
    - flex-2.5.31-debian_fixes-2.patch
    - man-1.5o1-80cols-1.patch
    - mktemp-1.5-add_tempfile-1.patch

- sysklogd-1.4.1-kernel_headers-1.patch
    - sysvinit-2.85-proclen-1.patch
    - texinfo-4.7-segfault-1.patch
    - util-linux-2.12b-sfdisk-1.patch
    - zlib-1.2.1-security-1.patch

- 9 de julho de 2005 [archaic]: Reescrito notas do kernel.
- 9 de julho de 2005 [matt]: Adicionada informação a respeito das listas de discussão de segurança e do freshmeat para chapter09/whatnow.xml. Bug 1583 corrigido. Agradecimentos a Steve Crosby pelo relatório e o texto sugerido.
- 7 de julho de 2005 [manuel]: Revisados os pacotes e tamanhos dos patchs. Usando o pacote lfs-packages-6.1.tar e o `du - k` para medi-lo. Corrigido a marca (tag) beginpage para a saída do PDF. Removidas as páginas em branco na saída do PDF para versões não publicadas.
- 6 de julho de 2005 [archaic]: Adicionado patch de segurança para a zlib.
- 6 de july de 2005 [matt]: Diversas correções de tipos, como sugerido por Bernard Leak.
- 5 de julho 2005 [archaic]: Removida referencia ao wiki. Apontada para o FAQ.
- 4 de julho de 2005 [archaic]: Reorganizada página de errata, assim referenciando somente avisos de segurança e correções de bugs, não mais novas características.
- 4 de julho de 2005 [archaic]: Todas (esperamos que sim) as referencias de páginas de man/info em conformidade. A conformidade se baseou no fato da referências ser à uma página específica do man ou à páginas do man em geral. Atualizada tipografia para refletir isto.
- 2 de julho de 2005 [archaic]: Várias mudanças pequenas nas frases nos capítulos 8 e 9 (matt). Também removido o parágrafo sobre a compressão de módulos do kernel porque está no material de dicas.
- 2 de julho de 2005 [archaic]: Pequenas alterações em frases do capitulo 8 (matt).
- 1 de julho de 2005 [archaic]: Pequenas alterações de palavras no capitulo 6 (matt).
- 1 de julho de 2005 [archaic]: Todas ocorrências de LFS-Bootscripts em conformidade.
- 30 de junho de 2005 [archaic]: Pequenas alterações de palavras nos capítulos de 1 ao 5 (matt).
- 30 de junho de 2005 [archaic]: Adicionada uma entidade para o livecd-root.
- 29 de junho de 2005 [archaic]: Movido a página com os requisitos do sistema anfitrião para o prefacio do livro.
- 28 de junho de 2005 [archaic]: Alterada a montagem do /dev no ramfs para o tmpfs.
- 27 de junho de 2005 [matthew]: Removida a menção de problemas na suite de testes do capítulo 1 já que informações mais compreensivas são dada no capítulo 5 (archaic).
- 27 de junho de 2005 [matthew]: Reformulada a descrição do caso da falha da atime da glibc e removida a descrição de falha do teste do shm visto que já montamos um tmpfs (archaic).

- 27 de junho de 2005 [archaic]: Preenchido o texto para página de errata. Grato pelo texto, Steve!
- 26 de junho de 2005 [manuel]: Pequenas correções de marcações (tags).
- 25 de junho de 2005 [archaic]: Adicionado um espaço reservado para a página de errata e um endereço eletrônico temporário (atualmente morto).
- 25 de junho de 2005 [archaic]: Adicionado entidades "generic-version" e "test-results".
- 25 de junho de 2005 [archaic]: Adicionado o vínculo simbólico compacto para o gzip.
- 25 de junho de 2005 [jhuntwork]: Adicionado o parâmetro --with-tclinclude na compilação do Expect build para garantir que ele saiba onde encontrar o diretório de fontes do Tcl.
- 25 de junho de 2005 [matthew]: Atualizada a última versão do patch do tempfile do mktemp a qual oferece suporte para a compilação fora do diretório de códigos fonte.
- 23 de junho de 2005 [archaic]: Reformulada a página do inputrc.
- 22 de junho de 2005 [archaic]: Adicionado endereço para os resultados dos testes.
- 22 de junho de 2005 [archaic]: Atualizado o Shadow para a versão 4.0.9. Removido o patch lastlog.
- 21 de junho de 2005 [archaic]: Removida a opção --with-included-regex, do capitulo05/grep, já que não parece ter uma razão valida para usa-la e a explicação dela estava incorreta.
- 21 de junho de 2005 [archaic]: Atualizado para o findutils-4.2.23.
- 20 de junho de 2005 [archaic]: Atualizado patch do Flex de -2 para -3.
- 20 de junho de 2005 [manuel]: Adicionado aviso sobre o os cabeçalhos do kernel e Linux-Libc-Headers, corrigida a lista de arquivos instalados no kernel.xml (bug 1569). Algumas correções de tipos e tags portados do trunk (r6048 ao r6050 e r6053 ao r6056.) Corrigida descrição do programa top (bug 1549.) Corrigida descrição do tar (bug 1553.) Reformulada explicação do patch Util-linux (bug 1554.)
- 19 de junho de 2005 [jhuntwork]: Alterada lista de servidores IRC para exibir somente irc.linuxfromscratch.org.
- 19 de junho de 2005 [jhuntwork]: Removida página desatualizada do bootcd e adicionada uma breve descrição do LiveCD à seção 1.1.
- 16 de junho de 2005 [archaic]: Adicionadas dependências de instalação para o hotplug.
- 16 de junho de 2005 [matthew]: Outros reparos de tipografia e marcação no capítulo 7, como reportado por Randy McMurchy.
- 16 de junho de 2005 [matthew]: Tipografia e marcações corrigidos no capítulo 7, por Randy McMurchy.
- 16 de junho de 2005 [jhuntwork]: Ajustada descrição do pacote de patch. Grato Randy McMurchy.
- 16 de junho de 2005 [archaic]: Corrigido link para página db BLFS's referenciado em iproute2 (mesclado de trunk r6006).
- 15 de junho de 2005 [archaic]: Adicionado --disable-nls para pass2 binutils para evitar a exigência de gettext (mesclado em trunk r5983).
- 14 de junho de 2005 [archaic]: Atualizado todos tamanhos de configuração (mesclado em r5916, r5917, r5918 e r5972).

- 14 de junho de 2005 [archaic]: Removido --with-included-regex do capitulo 6 uma vez que é menos confiável que glibc's em locales não-C.
- 14 de junho de 2005 [archaic]: Removidas referências aos tarballs separados do GCC (GCC-núcleo, gcc-g++ etc.)
- 12 de junho de 2005 [matt]: Atualizado para Linux 2.6.11.12.
- 8 de junho de 2005 [archaic]: Removida sugestão sobre onde mover /sources e reformulado o resto da página (chapter06/revisedchroot.xml).
- 8 de junho de 2005 [archaic]: Adicionado um comando para impedir que module-init-tools reescreva a página do man (o qual confia em docbook2man).
- 1 de junho de 2005 [manuel]: Patchs do root alterados para lfs/svn/testing/.
- 23 de maio de 2005 [manuel]: Pequenos melhoramentos de frases (agradecimentos à Peter Ennis).
- 22 de maio de 2005 [matt]: Atualizado para Linux 2.6.11.10.
- 15 de maio de 2005 [matt]: Adicionado patch de segurança.
- 15 de maio de 2005 [matt]: Atualizado para Linux 2.6.11.9.
- 15 de maio de 2005 [matt]: Atualizado para LFS-Bootscripts 3.2.1.
- 12 de maio de 2005 [matt]: Mais melhoramentos de frases e notas (agradecimentos à Peter Ennis e Tony Morgan)
- 12 de maio de 2005 [matt]: Pequenas melhorias de frases (agradecimentos à Peter Ennis)
- 27 de abril de 2005 [archaic]: Adicionado um patch que corrige duas falhas no testsuite quando rodando no kernel 2.6.11.x.
- 18 de abril de 2005 [manuel]: Ajustado o tag beginpage para equiparar às alterações da página anterior.
- 17 de abril de 2005 [manuel]: Atualizado o stylesheets para usar DocBook-XSL 1.68.1.
- 17 de abril de 2005 [matt]: Não cria arquivo de log de eventos do hotplug; o bootscripts manipula isto por nós.
- 17 de abril de 2005 [matt]: Usa charmaps canônico em /etc/profile e não ajusta LC_ALL (Ken Moffat e Alexander Patrakov)
- 16 de abril de 2005 [matt]: Reformula manipulação de dispositivos hotpluggable, agora que instalamos o pacote hotplug (Andrew Benton).
- 16 de abril de 2005 [matt]: Correção pequena de frases/tipos (Allard Welter).
- 16 de abril de 2005 [matt]: Correção pequena de frases/tipos (Peter Ennis).
- 16 de abril de 2005 [matt]: Removidas referências ao link estático do passo 1 do toolchain, o qual deve ter ido como parte do bug 1061 (Andrew Benton).
- 13 de abril de 2005 [manuel]: Correções ortográficas por Archiac. Adicionado tags para ajustar o PDF veja no capítulo 06.
- 12 de abril de 2005 [manuel]: Pequenas mudanças de redação. Adicionado tags para correção do PDF veja

em todos capítulos, exceto capítulo 06.

- 11 de abril de 2005 [manuel]: Menção ao testsuite do bzip2. Diversas tags e textos corrigidos.
- 6 de abril de 2005 [matt]: Movido comando sed e2fsprogs para antes de entrar a construção do diretório (Steffen R. Knollmann).
- 4 de abril de 2005 [matt]: Tipografia: O udev initscript registra o udevsend, não udev, como o manipulador hotplug (Bryan Kadzban).
- 4 de abril de 2005 [matt]: Não precisa criar manualmente `/var/log/hotplug` pois o Makefile do hotplug o cria (Ken Moffat). Também uma pequena reformulação para entendimento.
- 4 de abril de 2005 [matt]: Ajustado problema de compilação do e2fsprogs (Ken Moffat e Greg Schafer).
- 2 de abril de 2005 [jhuntwork]: Ajustado url dtd para o arquivo xml sysklogd.
- 31 de março de 2005 [jhuntwork]: Alterado o link para apontar para o menor ftp.gnu.org.
- 31 de março de 2005 [matt]: Atualizado para LFS-Bootscripts 3.2.0.
- 31 de março de 2005 [matt]: Atualizado para m4-1.4.3.
- 30 de março de 2005 [matt]: Atualizado para iproute2-2.6.11-050330.
- 30 de março de 2005 [jhuntwork]: Removido syslog-ng-1.6.6, libol-0.3.15. Reinstalado sysklogd-1.4.1. Agradecimentos à Archaic pelo patch.
- 26 de março de 2005 [matt]: Atualizado para linux-libc-headers-2.6.11.2
- 26 de março de 2005 [matt]: Atualizado para linux-libc-headers-2.6.11.1
- 26 de março de 2005 [matt]: Atualizado para linux-2.6.11.6
- 22 de março de 2005 [jim]: Atualizado para e2fsprogs-1.3.7.
- 21 de março de 2005 [jim]: Adicionado patch para corrigir problema com shadow e lastlog.
- 19 de março de 2005 [jim]: Adicionado patch para corrigir problema com tar -S
- 19 de março de 2005 [matt]: Removido referencia ao patch de segurança do kernel
- 19 de março de 2005 [jim]: Atualizado para udev-056
- 19 de março de 2005 [jim]: Atualizado para linux-2.6.11.5
- 19 de março de 2005 [jim]: Alterado referencia de Iproute2 para IPRoute2
- 18 de março de 2005 [jim]: Atualizado para Findutils 4.2.20
- 16 de março de 2005 [jim]: Atualizado para linux-2.6.11.4
- 16 de março de 2005 [jim]: Removido referencia ao patch de segurança do kernel
- 16 de março de 2005 [jim]: Removido patch find_update para IPRoute2, não é mais necessário
- 15 de março de 2005 [matt]: Atualizado para iproute2-2.6.11-050314
- 14 de março de 2005 [matt]: Lista de arquivos/diretorios instalados ordenados alfabeticamente pela descrição.

- 14 de março de 2005 [matt]: Correções de tipografia e reformulada a explicação do hotplug (assim esperamos) mais claramente
- 14 de março de 2005 [matt]: Atualizado para gettext-0.14.3
- 14 de março de 2005 [jim]: Adicionado `/var/log/hotplug` para capturar eventos do hotplug. Adicionado /lib/firmware para carregar firmware com o hotplug
- 13 de março de 2005 [jim]: Atualizado patch iproute2 db para iproute2-2.6.11-050310. Removido patch desnecessário find_update também para iproute2-2.6.11-050310
- 13 de março de 2005 [matt]: Atualizado para iproute2-2.6.11-050310
- 13 de março de 2005 [matt]: Atualizado para linux-2.6.11.3 e linux-libc-headers-2.6.11.0
- 13 de março de 2005 [matt]: Reformulado seção sobre SBUs para refletir a nova correção para o bug 1061
- 13 de março de 2005 [matt]: Link dinâmico do toolchain pass1 para o trabalho do bug 1061 e removido toda explicação relacionada ao texto.
- 12 de março de 2005 [matt]: Atualizado para udev-054
- 12 de março de 2005 [matt]: Atualizado para findutils-4.2.19
- 12 de março de 2005 [matt]: Atualizado para psmisc 21.6
- 10 de março de 2005 [matt]: gettext não mais instala libgettext{lib,src}.a (Jack Brown)
- 3 de março de 2005 [matt]: Removido --without-cvs das instruções do glibc, pois não mais usamos glibc CVS snapshots
- 3 de março de 2005 [matt]: Ajustado um par de tipos na área de donwloads.
- 2 de março de 2005 [matt]: Adicionado notas as características potenciais da versão do e2fsprogs em uma distribuição anfitriã. Corrigido bug 1047. Agradecimentos à Steve Crosby pelas sugestões de explicação do texto.
- 2 de março de 2005 [jim]: Atualizado área de downloads
- 28 de fevereiro de 2005 [jim]: Atualizado patch de correção do bash para -3
- 28 de fevereiro de 2005 [matt]: Atualizado binutils para 2.14.94.0.2.2
- 28 de fevereiro de 2005 [matt]: Alterado `/usr/bin/logger` para `/bin` pois o bootscripts precisa dele ali. Corrigido bug 1035.
- 28 de fevereiro de 2005 [matt]: Atualizado para iana-etc-1.04
- 28 de fevereiro de 2005 [matt]: Correção das instruções para invocar udev's testsuite (Randy McMurchy)
- 27 de fevereiro de 2005 [matt]: Correção do título do patch no capítulo 3. Corrigido bug 1049
- 27 de fevereiro de 2005 [matt]: Mencionado udev's testsuite. Corrigido bug 1042
- 27 de fevereiro de 2005 [matt]: Usa --without-csharp no lugar de --disable-csharp, pois o último não funciona como pretendido. Fixes bug 1033
- 27 de fevereiro de 2005 [matt]: Atualizado para gettext-0.14.2

- 27 de fevereiro de 2005 [matt]: Atualizado para findutils-4.2.18
- 27 de fevereiro de 2005 [matt]: Atualizado para bzip2-1.0.3
- 19 de fevereiro de 2005 [gerard]: Capitulo 5-Stripping: removido `doc` dos diretórios para ser removido em `/tools`. Este diretório não é mais criado.
- 19 de fevereiro de 2005 [jeremy]: Adicionado correção ao capítulo 5 construção do glibc para ajustar a desabilitação da funcionalidade selinux. Agradecimentos ao Bobson no IRC (bobson@bobson.net) por apontar esta saída. Fechado bugzilla 1034.
- 19 de fevereiro de 2005 [gerard]: Sincronizado ramo Testing com o corrente Unstable/Trunk. Movido o ramo Testing para Trunk e descontinuado o ramo Testing por lfs-dev discussão no ramo alterado.
- 5 de fevereiro de 2005 [matt]: Copiado arquivo pnp.distmap do hotplug para calar os avisos. Arrumado também algum texto explanatório
- 29 de janeiro de 2005 [matt]: Atualizado para sed-4.1.4
- 29 de janeiro de 2005 [matt]: Atualizado para procps-3.2.5
- 29 de janeiro de 2005 [matt]: Atualizado para shadow-4.0.7
- 29 de janeiro de 2005 [matt]: Atualizado para util-linux-2.12q.
- 27 de janeiro de 2005 [matt]: Adicionado um aviso que o symlink `/usr/src/linux` não deve ser criado. Corrigido bug 1012.
- 27 de janeiro de 2005 [matt]: Adicionado link para o local do ftp do live-cd. Corrigido bug 1014.
- 27 de janeiro de 2005 [matt]: Added bison, flex and m4 to binutils dependency list. Fixes Bug 1018.
- 27 de janeiro de 2005 [manuel]: Atualizado para gcc-3.4.3-specs-2.patch.
- 19 de janeiro de 2005 [jeremy]: Adicionado um symlink extra para libgcc_s.so para o capítulo 6 isto nunca foi migrado da versão instável até agora.
- 9 de janeiro de 2005 [matt]: Adicionado um patch de segurança para o kernel
- 9 de janeiro de 2005 [matt]: Adicionado um patch de segurança para o vim
- 9 de janeiro de 2005 [matt]: Atualizado para man-1.5p
- 9 de janeiro de 2005 [matt]: Atualizado para texinfo-4.8
- 9 de janeiro de 2005 [matt]: Atualizado para util-linux-2.12p
- 9 de janeiro de 2005 [matt]: Atualizado para udev-050
- 9 de janeiro de 2005 [matt]: Atualizado para tcl-8.4.9
- 9 de janeiro de 2005 [matt]: Atualizado para tar-1.15.1
- 9 de janeiro de 2005 [matt]: Atualizado para perl-5.8.6
- 9 de janeiro de 2005 [matt]: Atualizado para man-pages-2.01
- 9 de janeiro de 2005 [matt]: Atualizado para linux-libc-headers-2.6.10.0
- 9 de janeiro de 2005 [matt]: Atualizado para linux-2.6.10

- 9 de janeiro de 2005 [matt]: Atualizado para gcc-3.4.3
- 9 de janeiro de 2005 [matt]: Atualizado para bison-2.0
- 9 de janeiro de 2005 [matt]: Atualizado para autoconf-1.9.4
- 5 de janeiro de 2005 [jeremy]: Pequena correção textual na configuração de rede, pois iproute não reconheceria o velho formato eth0:1 para apelidos de IP. Fechado bug 1013.
- 5 de janeiro de 2005 [jeremy]: Adiconado o parametro --disable-selinux ao Capitulo 5 glibc. Permitindo construir de hosts com SELinux, como o Fedora Core 3
- 25 de dezembro de 2004 [jeremy]: Adicionado texto sugerido por MSB, fechando Bug 943
- 25 de dezembro de 2004 [jeremy]: Atualizado binutils para 2.14.94.0.2 deve corrigir problema com TLS strip que parecia, ao menos em X86
- 22 de dezembro de 2004 [manuel]: Readicionado ao chapter09/reboot.xml da version 5.1.
- 20 de dezembro de 2004 [manuel]: Feita localização do grub compatível com FHS.
- 19 de dezembro de 2004 [manuel]: Adicionado servidor IRC irc.lfs-matrix.de.
- 5 de dezembro de 2004 [jeremy]: Adicionado parâmetro DOCBOOKTOMAN ao Module-init-utils para o Module-init-utils - sem isso, a compilação falha. Obrigado Boris Buegling
- 2 de dezembro de 2004 [jeremy]: Removido o patch do bash display_wrap, a favor do novo patch de correção, e adicionado o patch avoid_WCONTINUED também.
- 2 de dezembro de 2004 [jeremy]: Atualizado para TCL 8.4.8, Grep 2.5.1a Util-linux 2.12i, Iana-etc 1.03, File 4.12, Module-init-tools 3.1, Procps 3.2.4
- 2 de dezembro de 2004 [jeremy]: Migrado as mudanças da versão instável para a construção do Glibc de encontro com linux-libc-headers no lugar de cabeçalhos do kernel cru, parecendo mais com que os desenvolvedores do kernel pensam que devem acontecer.
- 1 de dezembro de 2004 [jeremy]: Apagado Udev de ser construído no Capítulo 5, a favor de criar um mínimo conjunto de dispositivos no início do capítulo 6. Todos os dispositivos são criados depois da instalação do Udev próximo ao fim do capítulo 6
- 1 de dezembro de 2004 [jeremy]: Upgraded to Automake 1.9.3, Binutils 2.15.92.0.2, Findutils 4.2.3, GCC 3.4.2, Glibc 20041011, Iana-Etc 1.02 Iproute2 2.6.9-041019, LFS-Bootscripts 2.2.3, Libtool 1.5.10, Linux 2.6.9 Linux-libc-headers 2.6.9.1, Man 1.5o1, Man-pages 1.70, Shadow 4.0.6, Udev 046, Zlib 1.2.2, Hotplug 2004_09_23, Libol 0.3.14, Syslog-ng 1.6.5

Branch frozen for LFS 6.0 as of October 10, 2004

1.3. Suporte

1.3.1. FAQ

Se durante a configuração do sistema LFS você encontrar algum erro, tiver alguma pergunta, ou perceber alguma falha no livro, por favor consulte antes de qualquer coisa as páginas do FAQ (Frequently Asked Questions,) em *http://www.linuxfromscratch.org/faq/*.

1.3.2. Listas de discussão

O servidor `linuxfromscratch.org` hospeda um grande número de listas de discussão (mailing lists) utilizadas para o desenvolvimento do projeto LFS. Estas listas incluem o desenvolvimento principal e listas de suporte, entre outras.

Para mais informações sobre as diferentes listas, como subscrever, localização dos arquivos antigos, e outras informações adicionais, visite: *http://www.linuxfromscratch.org/mail.html*.

1.3.3. Servidor de notícias

As listas de discussão hospedadas em `linuxfromscratch.org` are são também acessíveis através do Network News Transfer Protocol (NNTP). Todas as mensagens enviadas a uma lista de discussão são copiadas no newsgroup correspondente, e vice-versa.

O servidor de notícia está situado em `news.linuxfromscratch.org`.

1.3.4. IRC

Diversos membros da comunidade LFS oferecem ajuda em nossa rede Internet Relay Chat (IRC). Antes de usar este suporte, por favor certifique-se antes de que sua pergunta não está respondida no FAQ do LFS ou nos arquivos das listas de discussão. Você pode encontrar a rede IRC da comunidade LFS em `irc.linuxfromscratch.org`. O nome do canal de suporte é #LFS-support.

1.3.5. Referências

Para a informações adicionais sobre os pacotes, muitas anotações úteis estão disponíveis na página de referências dos pacotes LFS situada em *http://www.linuxfromscratch.org/~matthew/LFS-references.html*.

1.3.6. Sites-espelho

O projeto LFS tem alguns sites-espelho (mirrors) que dão acesso ao conteúdo do website e disponibilizam o download dos pacotes utilizados de forma mais convenientes ao usuário. Visite por favor o website do projeto LFS em *http://www.linuxfromscratch.org/mirrors.html* para uma lista atual dos mirrors em atividade.

1.3.7. Informações de Contato

Dirija por favor todas as suas perguntas e comentários a uma das listas de discussão do LFS (veja acima).

1.4. Ajuda

Se você tiver alguma uma dúvida ou pergunta ao trabalhar com este livro, verifique primeiro a página do FAQ em *http://www.linuxfromscratch.org/faq/#generalfaq*. Geralmente as perguntas já foram respondidas lá. Se a sua ainda não tiver sido respondida, tente encontrar a origem do problema. A dica a seguir vai te dar uma orientação sobre como acabar com problemas: *http://www.linuxfromscratch.org/hints/downloads/files/errors.txt*.

Nós temos também uma comunidade LFS maravilhosa que está disposta oferecer o auxílio com as listas de discussão e IRC (veja a seção Section 1.3, “Suporte” deste livro). A fim ajudar a diagnosticar e resolver o problema, inclua por favor toda a informação relevante em seu pedido de ajuda.

1.4.1. Coisas a mencionar

Além de uma explanação breve do problema, é essencial incluir em todo pedido de ajuda:

- A versão do roteiro que está sendo usado (neste caso 6.1)
- A distribuição e a versão do anfitrião que está sendo usada criar LFS
- O pacote ou seção onde o problema foi encontrado
- A mensagem ou o sintoma exato de erro que está sendo recebido
- Se você se desviou das instruções exatas deste roteiro

Note

Desviar-se das instruções deste livro *não* significa que nós não vamos ajudar. Acima de tudo o projeto LFS é sobre a preferência pessoal. O alerta sobre alguns procedimentos estabelecidos fora dos padrões nos ajuda a avaliar e determinar possíveis causas de seu problema.

1.4.2. Problemas com o ./configure

Quando algo der errado durante o estágio em que o script **configure** for executado, verifique as últimas linhas do arquivo `config.log`. Este arquivo pode conter mensagens de erros encontrados durante o **configure** que não foram mostradas na tela. Inclua estas linhas *relevantes* se você decidir pedir ajuda.

1.4.3. Problemas na compilação

Para nos ajudar a encontrar a causa do problema, a saída na tela e o conteúdo de vários arquivos são bastante úteis. A saída do script **configure** e do comando **make** também podem ser úteis. Não inclua tudo o que aparecer, mas também não inclua pouco demais. Como exemplo, aqui está um trecho da saída do **make**:

```
gcc -DALIASPATH=\"/mnt/lfs/usr/share/locale:.\"
-DLOCALEDIR=\"/mnt/lfs/usr/share/locale\"
-DLIBDIR=\"/mnt/lfs/usr/lib\"
-DINCLUDEDIR=\"/mnt/lfs/usr/include\" -DHAVE_CONFIG_H -I. -I.
-g -O2 -c getopt1.c
gcc -g -O2 -static -o make ar.o arscan.o commands.o dir.o
expand.o file.o function.o getopt.o implicit.o job.o main.o
misc.o read.o remake.o rule.o signame.o variable.o vpath.o
default.o remote-stub.o version.o opt1.o
-lutil job.o: In function `load_too_high':
/lfs/tmp/make-3.79.1/job.c:1565: undefined reference
to `getloadavg'
collect2: ld returned 1 exit status
make[2]: *** [make] Error 1
make[2]: Leaving directory `/lfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/lfs/tmp/make-3.79.1'
make: *** [all-recursive-am] Error 2
```

Neste caso, muita gente incluiria apenas a parte debaixo, que começa por:

```
make [2]: *** [make] Error 1
```

Isto não é suficiente para diagnosticarmos o problema, porque apenas nos diz que "algo" deu errado, não *o quê* deu errado. A seção toda, como no exemplo acima, é o que deve ser incluso para ser útil, pois abrange o comando que foi executado e a(s) mensagem(ns) de erro apresentada.

Um excelente artigo sobre como perguntar por ajuda na Internet em geral foi escrito por Eric S. Raymond. Está disponível em *http://catb.org/~esr/faqs/smart-questions.html*. Leia e siga as dicas deste documento e você estará apto a receber as respostas e interpretá-las corretamente e também a achar a ajuda que realmente precisa.

Chapter 2. Preparando uma nova partição

2.1. Introdução

Neste capítulo, vamos preparar a partição que hospedará o sistema LFS. Vamos criar uma partição própria, um sistema de arquivos e vamos montá-la.

2.2. Criando uma nova partição

Como a maioria dos sistemas operacionais, o LFS é instalado geralmente em uma partição dedicada. É recomendado construir um sistema LFS em uma partição vazia disponível ou, se você tiver bastante espaço não particionado, criar uma. Entretanto, um sistema LFS (ou mesmo múltiplos sistemas LFS) pode também ser instalado em uma partição ocupada por um outro sistema operacional e os dois sistemas podem co-existir pacificamente. O documento *http://www.linuxfromscratch.org/hints/downloads/files/lfs_next_to_existing_systems.txt* explica como fazer isto, já que este roteiro aborda o método de usar uma partição nova para a instalação.

Um sistema mínimo requer uma partição de aproximadamente 1,3 gigabytes (GB). Isto é o bastante para armazenar todos os pacotes com os fontes e para compilá-los. Entretanto, se o sistema LFS vier a ser seu sistema Linux principal, o software adicional que será instalado provavelmente exigirá algum espaço adicional (2-3 GB). O próprio sistema LFS não fará uso de todo este espaço. Uma grande parcela desta exigência se deve à necessidade de fornecer espaço livre suficiente para armazenamento provisório. A compilação dos pacotes requer muito do espaço em disco, que é recuperado depois que o pacote é instalado.

Porque nunca temos memória de acesso aleatório (RAM) disponível para processos da compilação, é uma boa idéia usar uma partição pequena do disco como swap. Ela é usada pelo kernel para armazenar dados raramente usados e deixar mais memória física disponível para processos ativos. A partição swap para o sistema LFS pode ser a mesma que está sendo usada pelo sistema de anfitrião, e neste caso não é necessário criar outra.

Comece um programa de particionamento, tal como o **cfdisk** ou **fdisk** com uma opção de linha de comando que indique o disco rígido em que a partição nova será criada—por exemplo `/dev/hda` para o disco rígido de sua primeira controladora IDE. Crie uma partição nativa Linux e uma partição swap, se necessário. Consulte por favor as páginas do man `cfdisk(8)` ou `fdisk(8)` se você ainda não souber usar estes programas.

Memorize ou anote a designação da nova partição (por exemplo, `hda5`). Este livro irá se referir a ela tão somente como partição LFS. Memorize também a designação da partição swap. Estes nomes serão necessários mais tarde para a criação do arquivo `/etc/fstab`.

2.3. Criando um sistema de arquivos na partição

Agora que uma partição em branco foi criada, o sistema de arquivos pode ser criado. O sistema de arquivos mais usado no mundo Linux é o Second Extended File System (ext2), mas com os discos rígidos de alta capacidade mais novos, os sistemas de arquivo com journaling estão se tornando cada vez mais populares. Nós criaremos um sistema de arquivos ext2. As instruções da configuração para outros sistemas de arquivos podem ser encontradas em *http://www.linuxfromscratch.org/blfs/view/svn/ postlfs/filesystems.html*.

Para criar um sistema de arquivos ext2 na partição LFS, execute o seguinte comando:

```
mke2fs /dev/[xxx]
```

Substitua *[xxx]* pelo o nome da partição LFS (`hda5` em nosso exemplo).

Note

Algumas distribuições do sistema anfitrião usam características próprias em suas ferramentas da criação do sistema de arquivos (e2fsprogs). Isto pode causar problemas quando inicializar o seu novo sistema LFS no capítulo 9, porque aquelas características não serão suportadas pelo e2fsprogs instalado pelo LFS; você verá um erro similar a “unsupported filesystem features, upgrade your e2fsprogs”. Para verificar se seu sistema anfitrião utiliza alguma característica própria, execute o seguinte comando:

```
debugfs -R feature /dev/[xxx]
```

Se a saída obtida contiver características diferentes de dir_index; filetype; large_file; resize_inode or sparse_super então seu sistema anfitrião pode ter características próprias. Neste caso, para evitar problemas futuros, você deve compilar o pacote do e2fsprogs e usar os binários resultantes para recriar o sistema de arquivos em sua partição LFS:

```
cd /tmp
tar xjf /path/to/sources/e2fsprogs-1.37.tar.bz2
cd e2fsprogs-1.37
mkdir build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs /dev/[xxx]
cd /tmp
rm -rf e2fsprogs-1.37
```

Se uma partição swap for criada, ela precisa ser inicializada com o comando abaixo. Se você estiver usando uma partição swap já em uso pelo sistema anfitrião, não há nenhuma necessidade de formatá-la.

```
mkswap /dev/[yyy]
```

Substitua *[yyy]* pelo nome da partição swap.

2.4. Montando a nova partição

Agora que um sistema de arquivo foi criado, a partição precisa se tornar acessível. Para fazer isto, a partição precisa ser montada em um ponto de montagem escolhido. Para as finalidades deste livro, vamos supor que o sistema de arquivos será montado em `/mnt/lfs`, mas quem escolhe o diretório é você.

Escolha um ponto da montagem e atribua-o à variável de ambiente `LFS` com o comando:

```
export LFS=/mnt/lfs
```

Em seguida, crie o ponto da montagem e monte o sistema de arquivos do LFS com o comando:

```
mkdir -p $LFS
mount /dev/[xxx] $LFS
```

Substitua *[xxx]* pela designação da partição LFS.

Se usando partições múltiplas para o LFS (por exemplo, uma para `/` e outra para `/usr`), monte-as usando:

```
mkdir -p $LFS
mount /dev/[xxx] $LFS
mkdir $LFS/usr
mount /dev/[yyy] $LFS/usr
```

Substitua *[xxx]* e *[yyy]* pelos nomes apropriados da partição.

Assegure-se de que esta nova partição não seja montada com permissões muito restritivas (como nosuid, nodev, ou noatime). Use o comando **mount** sem nenhum parâmetro para ver quais opções foram definidas para a partição LFS montada. Se o *nosuid*, o *nodev*, e/ou o *noatime* estiverem definidos, a partição precisará ser remontada.

Agora que temos um lugar para trabalhar, é hora de fazer o download dos pacotes.

Part II. Preparando a configuração

Chapter 3. Pacotes e patches

3.1. Introdução

Este capítulo inclui uma lista dos pacotes que necessitam ser baixados da Internet (download) para a construção de um sistema básico Linux. Os números das versões listadas aqui correspondem às versões estáveis atualmente conhecidas destes softwares, e este livro está baseado nelas. Recomendamos não usar versões atualizadas porque os comandos para uma versão podem não funcionar da mesma forma com uma versão mais nova. As versões mais novas podem também apresentar problemas com dependências, e estas dependências serão desenvolvidas e estabilizadas nas próximas versões deste livro.

Os locais indicados para download podem não estar mais acessíveis. Se um site de download mudar após a publicação deste roteiro, o Google (*http://www.google.com/*) uma ferramenta de busca muito útil para a maioria dos pacotes. Se esta busca for mal sucedida, tente um dos meios alternativos para download discutidos em *http://www.linuxfromscratch.org/ lfs/packages.html*.

Os pacotes e os patches baixados precisam ser armazenados em algum lugar convenientemente disponível durante toda a configuração. Um diretório de trabalho é necessário também para descompactar as fontes e para a compilação. O diretório `$LFS/sources` pode ser usado como um lugar para armazenar os pacotes e patches e como diretório de trabalho. Usando este diretório, os elementos requeridos ficarão na partição LFS e estarão disponíveis durante todos os estágios do processo de configuração.

Para criar este diretório, execute, como usuário *root* o seguinte comando antes de começar a sessão de download:

```
mkdir $LFS/sources
```

Defina as permissões deste diretório como writable e sticky. “Sticky” significa que mesmo que todos os usuários tenham permissão de escrita em um diretório, apenas o proprietário de um arquivo poderá apagá-lo. O seguinte comando permitirá a escrita e o modo sticky:

```
chmod a+wt $LFS/sources
```

3.2. Todos os Pacotes

Faça o download ou obtenha de outra maneira os seguintes pacotes:

- Autoconf (2.59) - 908 kilobytes (KB):
  http://ftp.gnu.org/gnu/autoconf/
- Automake (1.9.5) - 748 KB:
  http://ftp.gnu.org/gnu/automake/
- Bash (3.0) - 1,824 KB:
  http://ftp.gnu.org/gnu/bash/
- Binutils (2.15.94.0.2.2) - 11,056 KB:
  http://www.kernel.org/pub/linux/devel/binutils/
- Bison (2.0) - 916 KB:
  http://ftp.gnu.org/gnu/bison/
- Bzip2 (1.0.3) - 596 KB:
  http://www.bzip.org/
- Coreutils (5.2.1) - 4,184 KB:
  http://ftp.gnu.org/gnu/coreutils/
- DejaGNU (1.4.4) - 852 KB:
  http://ftp.gnu.org/gnu/dejagnu/
- Diffutils (2.8.1) - 648 KB:
  http://ftp.gnu.org/gnu/diffutils/
- E2fsprogs (1.37) - 3,100 KB:
  http://prdownloads.sourceforge.net/e2fsprogs/
- Expect (5.43.0) - 416 KB:
  http://expect.nist.gov/src/
- File (4.13) - 324 KB:
  ftp://ftp.gw.com/mirrors/pub/unix/file/

Note

O File (4.13) pode não estar disponível por muito tempo neste site. Os administradores do site removem ocasionalmente as versões mais antigas quando as novas são liberadas. Um site alternativo para download que pode ter a versão correta disponível é *ftp://ftp.linuxfromscratch.org/pub/lfs/*.

- Findutils (4.2.23) - 784 KB:
  http://ftp.gnu.org/gnu/findutils/
- Flex (2.5.31) - 672 KB:
  http://prdownloads.sourceforge.net/lex/

- Gawk (3.1.4) - 1,696 KB:
  http://ftp.gnu.org/gnu/gawk/
- GCC (3.4.3) - 26,816 KB:
  http://ftp.gnu.org/gnu/gcc/
- Gettext (0.14.3) - 4,568 KB:
  http://ftp.gnu.org/gnu/gettext/
- Glibc (2.3.4) - 12,924 KB:
  http://ftp.gnu.org/gnu/glibc/
- Glibc-Linuxthreads (2.3.4) - 236 KB:
  http://ftp.gnu.org/gnu/glibc/
- Grep (2.5.1a) - 520 KB:
  http://ftp.gnu.org/gnu/grep/
- Groff (1.19.1) - 2,096 KB:
  http://ftp.gnu.org/gnu/groff/
- GRUB (0.96) - 768 KB:
  ftp://alpha.gnu.org/gnu/grub/
- Gzip (1.3.5) - 284 KB:
  ftp://alpha.gnu.org/gnu/gzip/
- Hotplug (2004_09_23) - 40 KB:
  http://www.kernel.org/pub/linux/utils/kernel/hotplug/
- Iana-Etc (1.04) - 176 KB:
  http://www.sethwklein.net/projects/iana-etc/downloads/
- Inetutils (1.4.2) - 752 KB:
  http://ftp.gnu.org/gnu/inetutils/
- IPRoute2 (2.6.11-050330) - 276 KB:
  http://developer.osdl.org/dev/iproute2/download/
- Kbd (1.12) - 624 KB:
  http://www.kernel.org/pub/linux/utils/kbd/
- Less (382) - 216 KB:
  http://ftp.gnu.org/gnu/less/
- LFS-Bootscripts (3.2.1) - 32 KB:
  http://downloads.linuxfromscratch.org/
- Libtool (1.5.14) - 1,604 KB:
  http://ftp.gnu.org/gnu/libtool/
- Linux (2.6.11.12) - 35,792 KB:
  http://www.kernel.org/pub/linux/kernel/v2.6/
- Linux-Libc-Headers (2.6.11.2) - 2,476 KB:
  http://ep09.pld-linux.org/~mmazur/linux-libc-headers/

- M4 (1.4.3) - 304 KB:
  http://ftp.gnu.org/gnu/m4/
- Make (3.80) - 904 KB:
  http://ftp.gnu.org/gnu/make/
- Man (1.5p) - 208 KB:
  http://www.kernel.org/pub/linux/utils/man/
- Man-pages (2.01) - 1,640 KB:
  http://www.kernel.org/pub/linux/docs/manpages/
- Mktemp (1.5) - 68 KB:
  ftp://ftp.mktemp.org/pub/mktemp/
- Module-Init-Tools (3.1) - 128 KB:
  http://www.kernel.org/pub/linux/utils/kernel/module-init-tools/
- Ncurses (5.4) - 1,556 KB:
  ftp://invisible-island.net/ncurses/
- Patch (2.5.4) - 156 KB:
  http://ftp.gnu.org/gnu/patch/
- Perl (5.8.6) - 9,484 KB:
  http://ftp.funet.fi/pub/CPAN/src/
- Procps (3.2.5) - 224 KB:
  http://procps.sourceforge.net/
- Psmisc (21.6) - 188 KB:
  http://prdownloads.sourceforge.net/psmisc/
- Readline (5.0) - 1,456 KB:
  http://ftp.gnu.org/gnu/readline/
- Sed (4.1.4) - 632 KB:
  http://ftp.gnu.org/gnu/sed/
- Shadow (4.0.9) - 1,084 KB:
  ftp://ftp.pld.org.pl/software/shadow/

Note

O Shadow (4.0.9) pode não estar disponível por muito tempo neste site. Os administradores do site removem ocasionalmente as versões mais antigas quando as novas são liberadas. Um site alternativo para download que pode ter a versão correta disponível é *ftp://ftp.linuxfromscratch.org/pub/lfs/*.

- Sysklogd (1.4.1) - 72 KB:
  http://www.infodrom.org/projects/sysklogd/download/
- Sysvinit (2.86) - 88 KB:
  ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/

- Tar (1.15.1) - 1,580 KB:
  http://ftp.gnu.org/gnu/tar/
- Tcl (8.4.9) - 2,748 KB:
  http://prdownloads.sourceforge.net/tcl/
- Texinfo (4.8) - 1,492 KB:
  http://ftp.gnu.org/gnu/texinfo/
- Udev (056) - 476 KB:
  http://www.kernel.org/pub/linux/utils/kernel/hotplug/
- Udev Rules Configuration - 5 KB:
  http://downloads.linuxfromscratch.org/udev-config-3.rules
- Util-linux (2.12q) - 1,344 KB:
  http://www.kernel.org/pub/linux/utils/util-linux/
- Vim (6.3) - 3,620 KB:
  ftp://ftp.vim.org/pub/vim/unix/
- Vim (6.3) language files (optional) - 540 KB:
  ftp://ftp.vim.org/pub/vim/extra/
- Zlib (1.2.2) - 368 KB:
  http://www.zlib.net/

Tamanho total destes pacotes: 146 MB

3.3. Patches necessários

Além dos pacotes, diversos patches também são necessários. Estes patches corrigem erros nos pacotes durante a configuração. Eles também fazem pequenas modificações que tornam os programas mais fáceis usar. Os seguintes patches serão necessários para montar o sistema LFS:

- Bash Various Fixes - 23 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/bash-3.0-fixes-3.patch
- Bash Avoid Wcontinued Patch - 1 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/bash-3.0-avoid_WCONTINUED-1.patch
- Coreutils Suppress Uptime, Kill, Su Patch - 15 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/coreutils-5.2.1-suppress_uptime_kill_su-1.patch
- Coreutils Uname Patch - 4 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/coreutils-5.2.1-uname-2.patch
- Expect Spawn Patch - 7 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/expect-5.43.0-spawn-1.patch
- Flex Brokenness Patch - 156 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/flex-2.5.31-debian_fixes-3.patch
- GCC Linkonce Patch - 12 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/gcc-3.4.3-linkonce-1.patch
- GCC No-Fixincludes Patch - 1 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/gcc-3.4.3-no_fixincludes-1.patch
- GCC Specs Patch - 14 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/gcc-3.4.3-specs-2.patch
- Glibc Fix Testsuite Patch - 1 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/glibc-2.3.4-fix_test-1.patch
- Gzip Security Patch - 2 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/gzip-1.3.5-security_fixes-1.patch
- Inetutils Kernel Headers Patch - 1 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/inetutils-1.4.2-kernel_headers-1.patch
- Inetutils No-Server-Man-Pages Patch - 4 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/inetutils-1.4.2-no_server_man_pages-1.patch
- IPRoute2 Disable DB Patch - 1 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/iproute2-2.6.11_050330-remove_db-1.patch
- Mktemp Tempfile Patch - 3 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/mktemp-1.5-add_tempfile-2.patch
- Perl Libc Patch - 1 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/perl-5.8.6-libc-1.patch

- Readline Fixes Patch - 7 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/readline-5.0-fixes-1.patch
- Sysklogd Fixes Patch - 27 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/sysklogd-1.4.1-fixes-1.patch
- Tar Sparse Fix Patch - 1 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/tar-1.15.1-sparse_fix-1.patch
- Util-linux Cramfs Patch - 3 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/util-linux-2.12q-cramfs-1.patch
- Vim Security Patch - 8 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/vim-6.3-security_fix-1.patch
- Zlib Security Patch - 1 KB:
  http://www.linuxfromscratch.org/patches/lfs/6.1/zlib-1.2.2-security_fix-1.patch

Além dos patches acima requeridos, existem patches opcionais criados pela comunidade LFS. Estes patches opcionais resolvem problemas menores ou permitem uma funcionalidade que não é oferecida pelo pacote original. Sinta-se à vontade e use a base de dados situada em *http://www.linuxfromscratch.org/patches/* para pegar todos os patches adicionais que sirvam às necessidades do seu sistema.

Chapter 4. Preparações Finais

4.1. Sobre a variável $LFS

Durante todo este livro, a variável de ambiente `LFS` utilizada diversas vezes. É essencial que esta variável esteja sempre definida. Ela deve conter o ponto de montagem escolhido para a partição LFS. Verifique se a variável `LFS` está ajustada corretamente com o comando:

```
echo $LFS
```

Certifique-se que a saída do comando indique o ponto da montagem definido para a partição LFS, que será `/mnt/lfs` se o exemplo dado tiver sido seguido. Se a saída estiver incorreta, a variável pode ser ajustada novamente com o comando:

```
export LFS=/mnt/lfs
```

Ter esta variável definida será útil por exemplo em comandos como **mkdir $LFS/tools**. O shell substituirá automaticamente o “$LFS” por “/mnt/lfs” (ou o que quer que a variável contenha) quando processar a linha de comando.

Não se esqueça de verificar se a variável `$LFS` está definida sempre que você reinicializar o ambiente atual (como ao fazer um “su” para *root* ou para outro usuário).

4.2. Criando o diretório $LFS/tools

Todos os programas compilados no Chapter 5 serão instalados no diretório `$LFS/tools` para que fiquem separados dos programas compilados no Chapter 6. Os programas compilados aqui são ferramentas provisórias e não serão parte do sistema final LFS. Mantendo estes programas em um diretório separado, eles poderão ser facilmente descartados mais tarde, após seu uso. Isto evita também que estes programas fiquem nos diretórios do sistema anfitrião (fácil de fazer por acidente no Chapter 5).

Crie o diretório com o seguinte comando, como *root*:

```
mkdir $LFS/tools
```

A etapa seguinte é criar o um vínculo simbólico (symlink) para o diretório `/tools` no sistema anfitrião. Ele apontará para o diretório recém-criado na partição LFS. Execute este comando como *root* também:

```
ln -s $LFS/tools /
```

Note

O comando acima está correto. O comando **ln** tem algumas variações sintáticas, portanto tome a precaução de consultar a documentação **info coreutils ln** e `ln(1)` antes de buscar por ajuda sobre algo que você pode pensar se tratar de um erro.

O vínculo simbólico (symlink) criado permite que o processo de compilação do jogo de ferramentas fique vinculado sempre ao diretório `/tools`, o que significa que o compilador, o assembler, e o linker funcionarão adequadamente tanto neste capítulo (quando nós ainda estivermos usando algumas ferramentas do sistema anfitrião) quanto no próximo (quando nós faremos um “chrooted” para a partição LFS).

4.3. Adicionando o usuário LFS

Quando entramos no sistema como *root*, cometer um único erro pode danificar ou destruir todo o sistema. Conseqüentemente, nós recomendamos configurar os pacotes neste capítulo como um usuário sem privilégios de administrador. Você poderia usar seu próprio nome do usuário, mas para tornar mais fácil de ajustar um ambiente de trabalho limpo, crie um novo usuário chamado *lfs* como membro de um novo grupo (também chamado *lfs*) e use este usuário durante o processo de instalação. Como *root*, execute os seguintes comandos para adicionar o novo usuário:

```
groupadd lfs
useradd -s /bin/bash -g lfs -m -k /dev/null lfs
```

Entenda as opções da linha de comando:

-s /bin/bash
: Faz o **bash** shell padrão para o usuário *lfs*.

-g lfs
: Esta opção adiciona o usuário *lfs* ao grupo *lfs*.

-m
: Cria um diretório home para o usuário *lfs*.

-k /dev/null
: Este parâmetro impede a cópia dos arquivos-modelo do diretório skeleton (normalmente é `/etc/skel`) alterando a posição da entrada padrão para o dispositivo especial.

lfs
: Este é o nome atual para o grupo e o usuário criados.

Para iniciar fazer o login como *lfs* (ao contrário de comutar para o usuário *lfs* quando se fez o login como *root*, que não exige que o usuário *lfs* tenha uma senha), vamos dar ao usuário *lfs* uma senha:

```
passwd lfs
```

Garanta ao usuário *lfs* acesso irrestrito ao diretório `$LFS/tools` fazendo-o proprietário do diretório:

```
chown lfs $LFS/tools
```

Se um diretório separado foi criado para trabalho, como sugerimos, faça o *lfs* dono desse diretório:

```
chown lfs $LFS/sources
```

Em seguida, vamos iniciar uma sessão como usuário *lfs*. Isto pode ser feito através de um console virtual, gerenciador de display, ou com o comando:

```
su - lfs
```

O parâmetro “`-`” instrui o **su** para começar um login-shell ao contrário de um non-login-shell. A diferença entre estes dois tipos de shell pode ser encontrada em detalhes na página em `bash(1)` e **info bash**.

4.4. Configurando o ambiente

Vamos definir um bom ambiente de trabalho criando dois arquivos de inicialização para o shell **bash**. Faça o login como usuário *lfs*, e execute o seguinte comando para criar um novo `.bash_profile`:

```
cat > ~/.bash_profile << "EOF"
exec env -i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash
EOF
```

Quando fizer o login como usuário *lfs*, o shell normalmente inicia como *login* shell que lê o arquivo `/etc/profile` do sistema anfitrião (provavelmente contendo alguns ajustes e variáveis de ambiente) e em seguida o `.bash_profile`. O comando **exec env -i.../bin/bash** no arquivo `.bash_profile` substitui o shell em execução por um novo shell com ambiente completamente vazio, à exceção das variáveis `HOME`, `TERM` e `PS1`. Isto nos assegura que nenhuma variável de ambiente não desejada e potencialmente perigosa do sistema anfitrião escape para o ambiente de configuração. A técnica usada aqui consegue o objetivo de assegurar um ambiente limpo.

Uma nova instância do shell é um *non-login*, que não lê os arquivos `/etc/profile` ou `.bash_profile`, mas lê o arquivo `.bashrc`Vamos criar o arquivo `.bashrc` agora:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL PATH
EOF
```

O comando **set +h** desliga a função hash do **bash**. Hashing é normalmente um recurso útil—o **bash** usa uma tabela para recordar o caminho de arquivos executáveis e evitar a procura repetida pela variável `PATH` para encontrar outra vez o mesmo executável. Entretanto, as novas ferramentas devem ser usadas assim que forem instaladas. Desligando a função hash, o shell vai sempre pesquisar o `PATH` quando um programa estiver para ser executado. Assim, o shell vai encontrar as ferramentas mais recentes, instaladas dentro do diretório de ferramentas `$LFS/tools` tão logo elas estejam disponíveis, sem recordar de outra versão do mesmo programa em uma localização diferente.

Ajustar as permissões para criação de arquivos (umask) para 022 assegura que os arquivos e diretórios criados tenham permissão de escrita somente por seu proprietário, mas tenham permissões de leitura e execução por qualquer um (por padrão os novos arquivos são criados com o modo de permissão 644 e os diretórios com o modo 755).

A variável `LFS` deve ser definida com o ponto de montagem escolhido.

A variável `LC_ALL` controla o localização de determinados programas, fazendo com que suas mensagens sigam as convenções de um determinado país. Se o sistema anfitrião usar uma versão da Glibc anterior a 2.2.4, ter a variável `LC_ALL` definida com algum valor que não "POSIX" ou "C" (durante este capítulo) pode causar problemas se você sair do ambiente e desejar retornar mais tarde. Ajustando `LC_ALL` para "POSIX" ou "C" (os dois são equivalentes) asseguramos que tudo trabalhe como esperado no ambiente chroot.

Colocar `/tools/bin` como primeiro diretório no `PATH` faz com que todos os programas instalado no Chapter 5 sejam usados pelo shell imediatamente após sua instalação. Isto, combinado com o desligamento da função

hash, elimina o risco de programas do sistema anfitrião serem usados quando os novos programas já estiverem disponíveis (no capítulo 5).

Finalmente, para termos um ambiente totalmente preparado para a configuração das ferramentas provisórias, vamos ativar o perfil de usuário recém-criado:

```
source ~/.bash_profile
```

4.5. SBUs

Muitas pessoas gostariam de saber de antemão quanto tempo aproximadamente demora para compilar e instalar cada pacote. Como um sistema LFS pode ser configurado em muitos sistemas diferentes, é impossível fornecer uma estimativa exata de tempo. O maior pacote (Glibc) demora aproximadamente 20 minutos nos sistemas os mais rápidos, mas pode demorar até três dias para compilar em alguns sistemas mais lentos! Em vez de fornecer tempos reais, uma medida padrão (SBU) será usada.

O SBU é medido da seguinte maneira. O primeiro pacote a ser compilado neste livro é o Binutils, no Chapter 5. O tempo de demora para compilar este pacote é o que será considerado como nossa unidade de medida, o SBU. O tempo de demora dos demais processos de compilação serão estimados com base neste tempo.

Por exemplo, considere um pacote cujo tempo de compilação seja 4,5 SBUs. Isto significa que se um sistema demorou 10 minutos para compilar e instalar a primeira versão do Binutils, demorará *aproximadamente* 45 minutos para compilar este outro pacote. Felizmente, na maioria de vezes a compilação demoram menos que para o Binutils.

Mas a medida SBUs não é exata porque depende de muitos fatores, incluindo a versão do GCC do sistema anfitrião. Em Multi-Processamento Simétrico (as máquinas SMP), os SBUs são ainda menos exatos. São fornecidos neste livro apenas para dar uma estimativa de quanto tempo pode demorar para instalar um pacote, mas os números podem variar em até dúzias de minutos em alguns casos.

Para conhecer os tempos reais obtidos em algumas configurações de hardware, veja o The LinuxFromScratch SBU Home Page, em *http://www.linuxfromscratch.org/~bdubbs/*.

4.6. Suites de testes

A maioria dos pacotes fornecem um conjunto de testes (test suite). Executar o conjunto de testes para um pacote recém-compilado é uma boa idéia porque fornece uma “verificação de bom funcionamento” indicando que tudo foi compilado corretamente. Uma suite de testes que faça a contento suas verificações prova geralmente que o pacote está funcionando como pretendido. Entretanto, não garante que o pacote está totalmente livre de erros.

Algumas suites de testes são mais importantes do que outros. Por exemplo, as suites de testes dos pacotes de ferramentas — GCC, Binutils e Glibc — são de enorme importância devido ao seu papel fundamental para que o sistema funcione corretamente. As suites de testes do GCC e da Glibc podem demorar um muito tempo para terminar seus testes, especialmente em uma máquina mais lenta, mas são imprescindíveis.

Note

A experiência nos mostra que há poucas vantagens em executar as suites de testes no Chapter 5. Não se pode negar o fato de que o sistema anfitrião exerce sempre alguma influência nos testes nesse capítulo, causando freqüentemente falhas inexplicáveis. Como as ferramentas compiladas no Chapter 5 são provisórias e descartadas posteriormente, nós não recomendamos executar as suites de testes neste capítulo. As instruções para executar as suites de testes são fornecidas, mas são estritamente opcionais.

Uma ocorrência comum é executar as suites de testes do Binutils e do GCC fora dos terminais PTYs. Isto pode resultar em um número elevado de falhas. Isto acontece por diversas razões, mas a causa mais provável é que o sistema anfitrião não tem o sistema de arquivos `devpts` ajustado corretamente. Esta ocorrência será discutida com mais detalhes no Chapter 5.

Às vezes as suites de testes falharão, mas por razões que os seus criadores estão cientes e julgaram não-críticas. Consulte os registros feitos em *http://www.linuxfromscratch.org/lfs/build-logs/6.1/* para verificar se estas falhas são ou não esperadas. Este endereço é válido para todos os testes durante todo este livro.

Chapter 5. Construindo um sistema provisório

5.1. Introdução

Este capítulo mostra como compilar e instalar um sistema Linux mínimo. Este sistema conterá apenas as ferramentas necessárias para começar a construir nosso sistema LFS no Chapter 6 e para garantir um ambiente de trabalho mais conveniente.

Há duas etapas em construir este sistema mínimo. A primeira é construir um jogo de ferramentas novo e independente do sistema anfitrião (compilador, assembler, linker, bibliotecas e alguns utilitários úteis). A segunda etapa usa este jogo de ferramentas para construir as demais ferramentas essenciais.

Os arquivos compilados neste capítulo serão instaladas no diretório `$LFS/tools` para mantê-los separados dos arquivos instalados no capítulo seguinte e dos arquivos da árvore de diretórios do sistema anfitrião. Como os pacotes compilados aqui são provisórios, nós não queremos que venham a "poluir" o nosso futuro sistema LFS.

Antes de configurar um pacote, ele deverá ser desempacotado pelo usuário *lfs* e o comando **cd** usado para entrar no diretório que será criado pelo próprio desempacotamento. Todas as instruções de configuração supõem que o shell utilizado é o **bash**.

Diversos pacotes serão corrigidos antes da compilação, mas somente quando o patch for necessário para corrigir algum problema. Muitas vezes o patch é necessário tanto neste como no próximo capítulo, mas às vezes é preciso apenas em um deles. Portanto, não se preocupe caso pareça estar faltando instruções para um determinado patch. Mensagens de advertência sobre *offset* ou *fuzz* podem também ser encontradas ao aplicar um patch. Não se preocupe com estes avisos, porque mesmo assim o patch foi aplicado com sucesso.

Durante a instalação dos pacotes, você verá diversas advertências do compilador aparecendo na tela. Essas advertências são normais e podem ser ignoradas. Elas são apenas o que dizem que são: advertências — a maioria sobre o uso impróprio, mas não ilegal, da sintaxe da linguagem C ou C++. Isso ocorre porque as normas técnicas do C mudam freqüentemente e alguns pacotes ainda utilizam a sintaxe anterior, o que não é realmente um problema.

Após instalar um pacote, apague os arquivos fonte e os diretórios criados durante a compilação, a menos que instruído de outra maneira. Suprimir as fontes conserva o espaço e impede a mistura de compilções quando o mesmo pacote for reinstalado mais tarde. Somente três dos pacotes precisam conservar seus códigos-fonte e diretórios de compilação para que seu conteúdo possa ser utilizado por comandos posteriores. Dê atenção especial a estes lembretes.

Verifique uma última vez se a variável de ambiente `LFS` está definida corretamente:

```
echo $LFS
```

Verifique se é exibido o caminho para o ponto de montagem da sua partição LFS, que deve ser `/mnt/lfs` se você seguiu o nosso exemplo.

5.2. Notas técnicas sobre as ferramentas provisórias

Esta seção explica algumas das razões e dos detalhes técnicos por trás do método de compilação utilizado. Não é necessário compreender agora tudo nesta seção. A maioria destas informações estarão mais fáceis de compreender após termos executado toda a compilação. Esta seção pode ser consultada a qualquer tempo.

O objetivo do Chapter 5 é fornecer um ambiente provisório que possa ser descartado posteriormente e a partir do qual seja possível fazer uma compilação limpa e livre de problemas, do sistema LFS no Chapter 6. Aqui nós vamos separar o sistema novo do sistema anfitrião tanto quanto possível e, ao fazermos isso, estaremos compilando um pacote de ferramentas autônomo e auto-suficiente. É importante destacar que todo o processo de compilação foi projetado para minimizar os riscos para leitores novos e fornecer ao mesmo tempo o máximo de valor educacional.

Important

Antes de continuar, procure saber o nome da sua plataforma de trabalho, freqüentemente chamado de target triplet. Na maioria das vezes será *i686-pc-linux-gnu*. Uma maneira simples determinar o target triplet do seu equipamento é executar o script **config.guess** que acompanha os códigos-fonte de muitos pacotes. Desempacote os códigos-fonte do Binutils, execute o script **`./config.guess`** e anote a saída.

Procure saber também o nome do vinculador dinâmico (dynamic linker) da sua plataforma, também chamado de carregador dinâmico (dynamic loader — para não ser confundido com o vinculador padrão **ld** que é parte do Binutils). O vinculador dinâmico que vem no pacote Glibc procura e carrega as bibliotecas compartilhadas necessárias para um determinado programa, prepara o programa para execução e então o executa. O nome do vinculador dinâmico normalmente será `ld-linux.so.2`. Em plataformas menos comuns, o nome poderá ser `ld.so.1` e nas novas plataformas de 64 bits podem ter nomes completamente diferentes. O nome do vinculador dinâmico da sua plataforma pode ser obtido diretamente no diretório `/lib` do sistema anfitrião. Um modo seguro de se determinar este nome será obtendo-o diretamente de um binário qualquer do sistema anfitrião com o comando: **`readelf -l <nome do binário> | grep interpreter`**, anotando então a saída. Uma referência com todas as plataformas autorizadas pode ser consultada no arquivo `shlib-versions` na raiz da árvore dos códigos fonte da Glibc.

Alguns dos elementos técnicos fundamentais sobre como funciona o método de configuração do Chapter 5:

- O processo é similar ao princípio da compilação cruzada, por meio do qual as ferramentas instaladas no mesmo conjunto trabalham em cooperação, fazendo assim a pequena “mágica” GNU
- A manipulação cuidadosa do caminho padrão de busca por bibliotecas do vinculador dinâmico assegura que os programas somente se vincularão com as bibliotecas escolhidas
- A manipulação cuidadosa do arquivo de especificações `specs` do **gcc** que diz ao compilador qual vinculador dinâmico será usado

O pacote Binutils é instalado primeiro porque script **configure** GCC e do Glibc executam vários testes no assembler e no vinculador dinâmico para determinar quais características destes programas serão habilitadas ou não. E é muito importante fazer isto logo no início. Um GCC, ou um Glibc, mal configurado pode resultar em um jogo de ferramentas defeituoso, cujo impacto poderá ser notado apenas no final da configuração do sistema.

Uma falha apontada pelo conjunto de testes revela logo o problema, evitando que mais trabalho seja desperdiçado.

O Binutils instala o assembler e o vinculador dinâmico em dois locais, `/tools/bin` e `/tools/$TARGET_TRIPLET/bin`. Ele estabelece um vínculo direto (hard link) entre ambos. Um aspecto importante do vinculador dinâmico é sua ordem de busca por bibliotecas. Esta informação, detalhada, pode ser obtida do comando **ld** quando utilizada a opção *--verbose*. Por exemplo, o comando **ld --verbose | grep SEARCH** nos mostrará os caminhos de busca atuais e a sua ordem. Ele nos mostrará quais arquivos são vinculados pelo **ld** compilando um programa modelo e passando a opção *--verbose* ao vinculador dinâmico. Por exemplo, **gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded** mostrará todos os arquivos abertos com sucesso durante a vinculação.

O pacote seguinte a ser instalado é o GCC. Um exemplo de o que pode ser visto na execução do **configure** é:

```
checking what assembler to use...
        /tools/i686-pc-linux-gnu/bin/as
checking what linker to use... /tools/i686-pc-linux-gnu/bin/ld
```

Isto é importante pelas razões já mencionadas. Demonstra também que o script de configuração do GCC não procura os diretórios do PATH para encontrar as ferramentas que vai usar. Entretanto, durante uma operação real do próprio **gcc**, os mesmos caminhos de busca não serão necessariamente utilizados. Para saber qual o vinculador padrão que o **gcc** utilizará, execute: **gcc -print-prog-name=ld**.

Uma informação detalhada pode ser obtida do **gcc** usando a opção *-v* na linha de comando ao compilar um programa qualquer. Para o exemplo, **gcc -v dummy.c** mostrará informação detalhada sobre o preprocessamento, a compilação, e os estágios do assembly, incluindo os caminhos de busca do **gcc**e sua ordem.

O próximo pacote a ser instalado é o Glibc. Os requisitos mais importantes para se configurar o Glibc são o compilador, as ferramentas binárias (binary tools), e os cabeçalhos do kernel. O compilador geralmente não é um problema porque o Glibc usará sempre o **gcc** ncontrado em um diretório do `PATH`. As ferramentas binárias e os cabeçalhos do kernel pode ser um pouco mais complicado. Desta forma, não corra nenhum risco e utilize as configurações disponíveis para se assegurar das escolhas corretas. Após rodar o **configure**, verifique o conteúdo do arquivo `config.make` em `glibc-build` para ver todos os detalhes importantes. Note o uso de *CC="gcc -B/tools/bin/"* para controlar quais ferramentas binárias serão usadas, e o uso das opções *-nostdinc* e *-isystem* que controlam o caminho de busca do compilador. Isto demonstra um aspecto importante do pacote Glibc—ele é auto-suficiente quanto aos mecanismos de configuração e geralmente não se socorre do conjunto de ferramentas padrão.

Após a instalação do Glibc, faça alguns ajustes para ter certeza de que as buscas e as vinculações ocorram somente dentro do diretório `/tools`. Instale um **ld** bem ajusatado, com um caminho de busca incorporado limitado ao diretório `/tools/lib`. Edite o arquivo de especificações do **gcc** para apontar para o novo vinculador dinâmico em `/tools/lib`. Esta última etapa é vital para o processo inteiro. Como visto, o caminho para o vinculador dinâmico será incorporado em cada arquivo executável (ELF - Executable and Link Format) compartilhado. Isto pode ser verificado com o comando **readelf -l <nome do binário> | grep interpreter**. Editar o arquivo de especificações do GCC nos assegura que cada programa compilado daqui até o fim deste capítulo vai usar o novo vinculador dinâmico em `/tools/lib`.

A necessidade usar o novo vinculador dinâmico é também a razão porque o patch Specs é aplicado na segunda passagem do GCC. Falhar ao fazer isto resulta no próprio programa GCC mantendo incorporado dentro de si o nome do vinculador dinâmico situado no diretório `/lib` do sistema anfitrião, o que afastaria nosso o objetivo de tornar o novo sistema independente do anfitrião.

Durante a segunda passagem de Binutils, nós poderemos utilizar a opção de configuração *--with-lib-path* para controlar caminho de busca por bibliotecas do **ld** Daqui em diante, o núcleo do jogo de ferramentas é

autônomo e independente. Os demais pacotes do Chapter 5 compilados com a nova Glibc em `/tools`.

Quando iniciarmos um ambiente com o chroot no Chapter 6, o principal pacote a ser instalado será o Glibc, devido à sua natureza auto-suficiente mencionada acima. Como este Glibc será instalado no diretório `/usr`, faremos um rápido ajuste nas configurações padrão do jogo de ferramentas, prosseguindo então com a configuração do sistema LFS.

5.3. Binutils-2.15.94.0.2.2 - primeira passagem

O pacote Binutils contém um vinculador dinâmico, um assembler, e outras ferramentas para manipular arquivos-objeto.

Tempo de compilação aproximado: 1.0 SBU
Espaço em disco necessário: 170 MB

Requisitos de instalação: Bash, Bison, Coreutils, Diffutils, Flex, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed e Texinfo

5.3.1. Instalação do Binutils

É importante que o Binutils seja o primeiro pacote compilado porque tanto o Glibc quanto o GCC executam vários testes no vinculador dinâmico e no assembler disponíveis para determinar qual de suas próprias características serão configuradas.

Este pacote é conhecido por apresentar problemas quando suas opções de otimização padrão (incluindo as opções *-march* e *-mcpu*) são modificadas. Se alguma variável de ambiente que modifique estas opções de otimização tiver sido definida, tais como `CFLAGS` e `CXXFLAGS`, remova-as quando for compilar este pacote.

A documentação deste pacote recomenda que sua configuração seja realizada em um diretório de trabalho diferente do diretório dos arquivos fonte:

```
mkdir ../binutils-build
cd ../binutils-build
```

Note

Para que os valores SBU utilizados no restante deste livro possam ter algum uso, meça o tempo que demora para configurar este pacote desde o início da configuração até o primeiro install. Para conseguir isto facilmente, coloque os três comandos que seguem dentro de um comando **time**desta forma: **`time { ./configure ... && make && make install; }`**.

Agora prepare o Binutils para a compilação:

```
../binutils-2.15.94.0.2.2/configure --prefix=/tools --disable-nls
```

Descrição das opções de configuração:

--prefix=/tools
Diz ao script de configuração para preparar-se para instalar os programas do Binutils no diretório `/tools`.

--disable-nls
Desabilita a internacionalização pois o i18n não é necessário para as ferramentas provisórias.

Continue com a compilação do pacote:

```
make
```

A compilação está agora completa. Normalmente nós executaríamos agora o conjunto de testes, mas neste

estágio inicial em que estamos a estrutura necessária para a execução dos testes (Tcl, Expect, e DejaGNU) ainda não está no seu devido lugar. Os benefícios de executar os testes neste momento são mínimos uma vez que os programas desta primeira passagem serão substituídos logo.

Instale o pacote:

```
make install
```

Em seguida, prepare o vinculador dinâmico a ser instalado somente na fase de “ajustes” que virá depois:

```
make -C ld clean
make -C ld LIB_PATH=/tools/lib
```

Descrição das opções de configuração:

-C ld clean
: Diz ao make para remover todos os arquivos compilados do subdiretório `ld`.

-C ld LIB_PATH=/tools/lib
: Esta opção reconfigura tudo no subdiretório `ld`. Especificar a variável `LIB_PATH` do Makefile na linha de comando permite que nós cancelemos o valor padrão e indiquemos a posição das ferramentas provisórias. O valor desta variável determina o caminho de busca da biblioteca padrão do vinculador dinâmico. Esta preparação será usada mais tarde neste capítulo.

Warning

Não remova os diretórios de configuração e os arquivos fonte do Binutils ainda. Eles serão necessários em seu estado atual mais tarde neste capítulo.

Detalhes deste pacote estão localizados na Section 6.13.2, “Conteúdo do Binutils.”

5.4. GCC-3.4.3 - primeira passagem

O pacote do GCC contém uma coleção de compiladores GNU, que inclui os compiladores C e C++.

Tempo de compilação aproximado: 4.4 SBU
Espaço em disco necessário: 219 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, Gettext, Glibc, Grep, Make, Perl, Sed e Texinfo

5.4.1. Instalação do GCC

Este pacote é conhecido por apresentar problemas quando suas opções de otimização padrão (incluindo as opções *-march* e *-mcpu*) são modificadas. Se alguma variável de ambiente que modifique estas opções de otimização tiver sido definida, tais como `CFLAGS` e `CXXFLAGS`, remova-as quando for compilar este pacote.

A documentação deste pacote recomenda que sua configuração seja realizada em um diretório de trabalho diferente do diretório dos arquivos fonte:

```
mkdir ../gcc-build
cd ../gcc-build
```

Prepare o GCC para compilação:

```
../gcc-3.4.3/configure --prefix=/tools \
    --libexecdir=/tools/lib --with-local-prefix=/tools \
    --disable-nls --enable-shared --enable-languages=c
```

Descrição das opções de configuração:

--with-local-prefix=/tools
Remove o diretório `/usr/local/include` do caminho de busca do **gcc**. Isto não é absolutamente essencial, entretanto ajuda a minimizar a influência do sistema anfitrião.

--enable-shared
Permite a configuração das bibliotecas `libgcc_s.so.1` e `libgcc_eh.a`. Ter a `libgcc_eh.a` disponível garante que o script de configuração do Glibc (o próximo pacote que nós vamos compilar) produza os resultados apropriados.

--enable-languages=c
Assegura que somente o compilador de C seja configurado.

Continue com a compilação do pacote:

```
make bootstrap
```

Descrição das opções de configuração:

bootstrap

Este parâmetro não apenas compila o GCC, mas o faz diversas vezes. Usa os programas compilados em um primeiro tempo para compilar-se uma segunda vez, e então uma terceira vez. Compara então esta segunda com esta terceira compilação para certificar-se que pode se remontar perfeitamente. Isto significa também que foi compilado corretamente.

A compilação está completa. Neste momento, o conjunto de testes está funcionado normalmente, mas, como mencionado antes, a estrutura necessária para a execução dos testes não está no lugar ainda. Os benefícios de executar os testes neste momento são mínimos, já que os programas desta primeira passagem serão substituídos logo.

Instale o pacote:

```
make install
```

Por fim, faça um vínculo simbólico. Muitos programas e scripts executam o **cc** ao invés de **gcc**, o qual é usado para manter programas genéricos e conseqüentemente úteis em todos os tipos dos sistemas de UNIX onde o compilador GNU C não esteja instalado. Executar o **cc** deixa o administrador de sistema livre para decidir qual compilador C instalar.

```
ln -s gcc /tools/bin/cc
```

Detalhes deste pacote estão na Section 6.14.2, “Conteúdo do GCC.”

5.5. Linux-Libc-Headers-2.6.11.2

O pacote Linux-Libc-Headers contém os cabeçalhos do kernel “organizados”.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 26.9 MB

Requisitos de instalação: Coreutils

5.5.1. Instalação do Linux-Libc-Headers

Por anos foi prática comum usar os cabeçalhos do kernel em seu estado “natural” direto do tarball do kernel) no diretório `/usr/include`, mas nos últimos anos, os desenvolvedores do kernel tiveram dificuldades inesperadas. Isto fez surgir o projeto Linux-Libc-Headers, que foi projetado para manter uma versão estável da Application Programming Interface (API) dos cabeçalhos do kernel Linux.

Instale os arquivos de cabeçalho:

```
cp -R include/asm-i386 /tools/include/asm
cp -R include/linux /tools/include
```

Se sua arquitetura não for i386 (ou compatível), ajuste o primeiro comando de acordo.

Detalhes deste pacote estão na Section 6.9.2, “Conteúdo do Linux-Libc-Headers.”

5.6. Glibc-2.3.4

O pacote de Glibc contém a biblioteca C principal. Esta biblioteca fornece as rotinas básicas de alocação de memória, busca em diretórios, abertura e fechamento de arquivos, leitura e escrita de arquivos, manipulação de strings, "pattern matching", aritmética e assim por diante.

Tempo de compilação aproximado: 11.8 SBU
Espaço em disco necessário: 454 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Make, Perl, Sed e Texinfo

5.6.1. Instalação do Glibc

Este pacote é conhecido por apresentar problemas quando suas opções de otimização padrão (incluindo as opções *-march* e *-mcpu*) são modificadas. Se alguma variável de ambiente que modifique estas opções de otimização tiver sido definida, tais como `CFLAGS` e `CXXFLAGS`, remova-as quando for compilar este pacote.

Importante ressaltar que compilar o Glibc de modo diverso do sugerido neste livro põe em risco a estabilidade do sistema.

O Glibc tem dois testes que falham quando o kernel em execução é o 2.6.11.x. Constatou-se que o problema estava relacionado com os próprios testes, e não com o libc nem com o kernel. Se você planeja executar os testes aplique este patch:

```
patch -Np1 -i ../glibc-2.3.4-fix_test-1.patch
```

A documentação deste pacote recomenda que sua configuração seja realizada em um diretório de trabalho diferente do diretório dos arquivos fonte:

```
mkdir ../glibc-build
cd ../glibc-build
```

Em seguida, prepare a Glibc o para a compilação:

```
../glibc-2.3.4/configure --prefix=/tools \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.0 --with-binutils=/tools/bin \
    --without-gd --with-headers=/tools/include \
    --without-selinux
```

Descrição das opções de configuração:

--disable-profile
: Compila as bibliotecas sem informações de perfil. Omita esta opção se o perfil das ferramentas provisórias for necessário.

--enable-add-ons
: Usa o add-on NPTL como biblioteca de threading do Glibc.

--enable-kernel=2.6.0
: Compila a biblioteca com suporte para o kernel Linux 2.6.x.

--with-binutils=/tools/bin

Embora não requerida, esta opção assegura que não haja nenhum erro pertinente aos programas do pacote Binutils usados durante a configuração do Glibc.

--without-gd

Impede a configuração do programa **memusagestat**, que insiste em fazer um vínculo com as bibliotecas do sistema anfitrião (libgd, libpng, libz etc.).

--with-headers=/tools/include

Compila o Glibc conforme os cabeçalhos instalados recentemente no diretório tools, de modo que ele saiba exatamente quais características o kernel tem para otimizar sua própria configuração.

--without-selinux

Quando configurando a partir de sistemas anfitriões que incluem a funcionalidade SELinux (por exemplo o Fedora Core 3), o Glibc seria configurado com sustentação para SELinux. Como o ambiente das ferramentas do LFS não tem suporte para SELinux, um Glibc compilado com este recurso não funcionará corretamente.

Durante este estágio o seguinte aviso pode aparecer:

```
configure: WARNING:
*** These auxiliary programs are missing or
*** incompatible versions: msgfmt
*** some features will be disabled.
*** Check the INSTALL file for required versions.
```

O programa **msgfmt** faltado ou incompatível geralmente não é grave, mas pode às vezes causar problemas durante a execução do conjunto de testes. Este programa **msgfmt** é parte do pacote de Gettext que o sistema anfitrião deve fornecer. Se o **msgfmt** está presente mas é incompatível, atualize o pacote de Gettext de sistema de anfitrião ou continue sem ele e veja se o conjunto de testes funciona sem problemas mesmo assim.

Compile o pacote:

```
make
```

A compilação está completa. Como foi dito, executar o conjunto de testes para as ferramentas provisórias instaladas neste capítulo não é essencial. Para executar o teste do Glibc (se quiser), utilize o seguinte comando:

```
make check
```

Para uma discussão das falhas do teste que são de maior importância, veja por favor a Section 6.11, “Glibc-2.3.4.”

Neste capítulo, alguns testes podem ser afetados negativamente por ferramentas existentes ou problemas no ambiente do sistema de anfitrião. As falhas nos testes do Glibc neste capítulo não são tipicamente preocupantes. O Glibc instalado no Chapter 6 é que será usado no final de tudo, de modo que este sim necessita passar pela maioria dos testes (mesmo no Chapter 6, algumas falhas podem ocorrer, por exemplo, com os testes do math).

Ao encontrar uma falha, faça uma anotação dela, e continue executando o comando **make check**. O conjunto de testes deve retomar de onde parou e continuar. Esta seqüência de para-e-continua pode ser contornada com o comando **make -k check**. Usando esta opção, esteja certo de registrar a saída de modo que o arquivo de registro possa ser examinado mais tarde.

Durante a instalação o Glibc emitirá um aviso de advertência sobre a ausência do arquivo `/tools/etc/ld.so.conf`. Evite este aviso com os comandos:

```
mkdir /tools/etc
touch /tools/etc/ld.so.conf
```

Instale o pacote:

```
make install
```

Países e culturas diferentes têm convenções diferentes sobre como comunicar-se. Estas convenções variam do formato para representar datas até algumas situações mais complexas, tais como a língua falada. A “internacionalização” dos programas GNU é executada pelo locale.

Note

Se a suite de testes não estivere sendo executada neste capítulo (como foi recomendado), não há nenhuma necessidade instalar agora os locales. Os locales apropriados serão instalados no capítulo seguinte.

Para instalar mesmo assim os locales do Glibc, use o seguinte comando:

```
make localedata/install-locales
```

Para economizar tempo, uma alternativa é executar comando precedente (que gera e instala cada locale Glibc) para instalar somente aqueles locales que são necessários. Isto pode ser feito usando o comando **localedef**. Informações sobre este comando são encontradas no arquivo `INSTALL` nos fontes do Glibc. Entretanto, alguns locales são essenciais para que os futuros pacotes passem nos testes, em especial os testes *libstdc++* do GCC. As seguintes instruções, em vez do *`install-locales`* acima, instalarão o conjunto mínimo dos locales necessários para que os testes funcionem com sucesso:

```
mkdir -p /tools/lib/locale
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i ja_JP -f EUC-JP ja_JP
```

Detalhes deste pacote estão na Section 6.11.4, “Conteúdo do Glibc.”

5.7. Ajustando as ferramentas provisórias

Agora que as bibliotecas C provisórias estão instaladas, todas as ferramentas compiladas neste capítulo deverão estar vinculadas ser a estas bibliotecas. Para fazer isto, o vinculador dinâmico e o arquivo de especificações do compilador precisam ser ajustados.

O vinculador dinâmico, preparado no final da primeira passagem do Binutils, mas não instalado, será instalado agora executando o seguinte comando dentro do diretório de trabalho do binutils `binutils-build`:

```
make -C ld install
```

Daqui em diante, tudo será vinculado exclusivamente às bibliotecas em `/tools/lib`.

Note

Se você não prestou atenção no aviso anterior para não descartar as fontes e o diretório de trabalho do Binutils na primeira passagem, ignore o comando acima. Existe a possibilidade de os programas configurados daqui em diante se ligarem às bibliotecas do sistema anfitrião. Isto não é o ideal, mas não é um grande problema. A situação será corrigida quando a segunda passagem do Binutils for instalada mais tarde.

Agora que o vinculador dinâmico está instalado e ajustado, os diretórios de códigos-fontes e de compilação do Binutils devem ser removidos.

A tarefa seguinte é modificar o arquivo de especificações do GCC de modo que aponte para o novo vinculador dinâmico. Um simples script sed fará isto:

```
SPECFILE=`gcc --print-file specs` &&
sed 's@ /lib/ld-linux.so.2@ /tools/lib/ld-linux.so.2@g' \
    $SPECFILE > tempspecfile &&
mv -f tempspecfile $SPECFILE &&
unset SPECFILE
```

Alternativamente, o arquivo de especificações pode ser editado à mão. Isto é feito substituindo cada ocorrência de “/lib/ld-linux.so.2” por “/tools/lib/ld-linux.so.2”

Verifique diretamente o conteúdo do arquivo de especificações para se certificar de que as mudanças pretendidas foram feitas.

Important

Se você está trabalhando em uma máquina onde o nome do vinculador dinâmico seja algo que não `ld-linux.so.2`, substitua “ld-linux.so.2” pelo o nome do vinculador dinâmico da sua máquina nos comandos acima. Consulte Section 5.2, “Notas técnicas sobre as ferramentas provisórias,” se necessário.

É possível que alguns arquivos incluídos do sistema anfitrião encontrem sua maneira de entrar no diretório privado de inclusão do GCC. Esta invasão acontece em conseqüência dos recursos de correção de inclusões “fixincludes”, do GCC, que faz parte do processo de compilação do pacote. Tudo será explicado, com mais detalhes, mais adiante, ainda neste capítulo. Execute o seguinte comando para eliminar esta possibilidade:

```
rm -f /tools/lib/gcc/*/*/include/{pthread.h,bits/sigthread.h}
```

Caution

Neste momento, é fundamental parar com a compilação para termos certeza de que as funções básicas (compilação e vinculação) do novo jogo de ferramentas estão funcionando como esperado. Para executar uma verificação de sanidade, execute os seguintes comandos:

```
echo 'main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /tools'
```

Se tudo estiver funcionando corretamente, não deve haver nenhuma mensagem de erro e a saída do último comando terá o formato:

```
[Requesting program interpreter:
    /tools/lib/ld-linux.so.2]
```

Note que `/tools/lib` aparece como prefixo do vinculador dinâmico.

Se a saída não for esta, ou não houver nenhuma saída, então algo está errado. Investigue e percorra novamente todas as etapas para encontrar onde está o problema para corrigi-lo. Esta situação deve ser resolvida antes de continuarmos. Primeiro, execute a verificação de sanidade outra vez, usando o **gcc** ao invés do **cc**. Se isto funcionar, então o vínculo simbólico `/tools/bin/cc` está faltando. Reveja a Section 5.4, “GCC-3.4.3 - primeira passagem” e instale o vínculo. Em seguida, assegure-se de que a variável `PATH` está corretamente definida. Isto pode ser verificado pelo comando **echo $PATH** e verificando se `/tools/bin` é o primeiro diretório da lista. Se o erro estiver no `PATH`, poder significar que você não está logado como usuário *lfs* ou algo nele foi mal ajustado na Section 4.4, “Configurando o ambiente”. Uma outra opção é que algo pode ter acontecido de errado com a modificação do arquivo de especificações, feita acima. Neste caso, refaça estas modificações, copiando-e-colando cuidadosamente os comandos.

Quando estiver tudo bem, apague os arquivos utilizados no teste:

```
rm dummy.c a.out
```

5.8. Tcl-8.4.9

O pacote Tcl contém as ferramentas da linguagem de comando (Tool Command Language).

Tempo de compilação aproximado: 0.9 SBU
Espaço em disco necessário: 23.3 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, e Sed

5.8.1. Instalação do Tcl

Este pacote e os dois seguintes (Expect e DejaGNU) são instalados para dar suporte à execução dos conjuntos de testes do GCC e do Binutils. Instalar três pacotes com a finalidades de fazer testes pode parecer excessivo, mas é tranqüilizador, se não for fundamental, saber que as ferramentas mais importantes estão funcionando corretamente. Mesmo que os conjuntos de testes não sejam executados neste capítulo (não são essenciais), estes pacotes serão necessários para a execução destes testes no Chapter 6.

Prepare o Tcl para a compilação:

```
cd unix
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Para testar os resultados: **`TZ=UTC make test`**. O conjunto de testes do Tcl apresenta falhas conhecidas quando sob determinadas condições do sistema anfitrião que não foram ainda inteiramente identificadas. Conseqüentemente, falhas nos testes aqui não causam surpresa e não são consideradas críticas. O parâmetro *`TZ=UTC`* ajusta o tempo do sistema ao Coordinated Universal Time (UTC), conhecido também como Greenwich Mean Time (GMT), mas somente durante a execução dos testes. Isto garante que os testes do relógio sejam feitos corretamente. Os detalhes da variável de ambiente `TZ` serão discutidos no Chapter 7.

Instale o pacote:

```
make install
```

Warning

Não remova o diretório dos fontes do `tcl8.4.9` ainda, porque o pacote seguinte precisará de seus cabeçalhos internos.

Defina uma variável contendo o caminho completo do diretório atual. O pacote seguinte, Expect, usará esta variável encontrar os cabeçalhos do Tcl.

```
cd ..
export TCLPATH=`pwd`
```

Faça agora uma ligação simbólica:

```
ln -s tclsh8.4 /tools/bin/tclsh
```

5.8.2. Conteúdo do Tcl

Programas instalados: tclsh (link to tclsh8.4) and tclsh8.4
Biblioteca instalada: libtcl8.4.so

Descrição rápida

tclsh8.4	O shell de comandos do Tcl
tclsh	Um vínculo para o tclsh8.4
`libtcl8.4.so`	A biblioteca do Tcl

5.9. Expect-5.43.0

O pacote Expect contém um programa para dialogar com outros programas interativos, obedecendo a um script predefinido.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 4.0 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Sed, e Tcl

5.9.1. Instalação do Expect

Primeiro, conserte um erro que pode resultar em falhas falsas durante a execução do conjunto de testes do GCC:

```
patch -Np1 -i ../expect-5.43.0-spawn-1.patch
```

Prepare o pacote para a compilação:

```
./configure --prefix=/tools --with-tcl=/tools/lib \
   --with-tclinclude=$TCLPATH --with-x=no
```

Descrição das opções de configuração:

--with-tcl=/tools/lib
Assegura que o script de configuração encontre a instalação do Tcl na posição provisória das ferramentas em vez de possivelmente encontrar a versão existente no sistema anfitrião.

--with-tclinclude=$TCLPATH
Diz explicitamente ao Expect onde encontrar o diretório das fontes do Tcl e seus cabeçalhos internos. Usar esta opção evita que o **configure** falhe porque não pode descobrir automaticamente a posição do diretório das fontes do Tcl.

--with-x=no
Diz ao script de configuração para não procurar pelo Tk (o componente GUI do Tcl) ou pelas bibliotecas do sistema de janelas X, ambos podem existir no sistema anfitrião mas não existem no ambiente provisório.

Compile o pacote:

```
make
```

Para testar os resultados: **make test**. O conjunto de testes do Expect falhas conhecidas sob determinadas condições do sistema anfitrião que não estão sob o nosso controle. Conseqüentemente, falhas nos testes neste momento não devem surpreender e não são consideradas críticas.

Instale o pacote:

```
make SCRIPTS="" install
```

Descrição das opções de configuração:

SCRIPTS=""
Impede a instalação dos scripts suplementares do Expect, que não são necessários.

Remova agora a variável `TCLPATH`:

```
unset TCLPATH
```

Os diretórios das fontes do Tcl e do Expect podem ser removidos agora.

5.9.2. Conteúdo do Expect

Programa instalado: expect
Biblioteca instalada: libexpect-5.42.a

Descrição rápida

expect	Comunica-se com outros programas interativos obedecendo a um script
`libexpect-5.42.a`	Contém as funções que permitem ao Expect ser utilizado como uma extensão do Tcl ou ser utilizado diretamente pelo C ou pelo C++ (sem o Tcl)

5.10. DejaGNU-1.4.4

O pacote de DejaGNU contém um sistema para testar outros programas.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 6.1 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, e Sed

5.10.1. Instalação do DejaGNU

Prepare o DejaGNU para a compilação:

```
./configure --prefix=/tools
```

Compile e instale o pacote:

```
make install
```

5.10.2. Conteúdo do DejaGNU

Programa instalado: runtest

Descrição rápida

runtest Um script que localiza o shell do **expect** e então executa o DejaGNU

5.11. GCC-3.4.3 - Pass 2

Tempo de compilação aproximado: 11.0 SBU
Espaço em disco necessário: 292 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, Gettext, Glibc, Grep, Make, Perl, Sed e Texinfo

5.11.1. Reinstalação do GCC

Este pacote é conhecido por apresentar problemas quando suas opções de otimização padrão (incluindo as opções *-march* e *-mcpu*) são modificadas. Se alguma variável de ambiente que modifique estas opções de otimização tiver sido definida, tais como `CFLAGS` e `CXXFLAGS`, remova-as quando for compilar este pacote.

As ferramentas necessárias para testar o GCC e o Binutils — Tcl, Expect e DejaGNU — estão instaladas agora. O GCC e o Binutils podem ser recompilados e vinculados à nova Glibc e testando corretamente (se executarmos os testes neste capítulo). Note que estes conjuntos de testes são altamente dependentes do funcionamento adequado dos PTYs, que são fornecidos pelo sistema anfitrião. Os PTYs são geralmente implementados através do sistema de arquivos `devpts`. Verifique se o sistema anfitrião está corretamente configurado quanto a isto executando um teste rápido:

```
expect -c "spawn ls"
```

A resposta pode ser:

```
The system has no more ptys.
Ask your system administrator to create more.
```

Se esta mensagem for recebida, o sistema anfitrião não tem seus PTYs configurados corretamente. Neste caso, não há como executar os testes para o GCC e o Binutils até que esta situação seja resolvida. Consulte o FAQ do LFS, em *http://www.linuxfromscratch.org//lfs/faq.html#no-ptys*, para mais informação sobre como resolver isto.

Primeiro, corrija um problema conhecido e faça um ajuste essencial:

```
patch -Np1 -i ../gcc-3.4.3-no_fixincludes-1.patch
patch -Np1 -i ../gcc-3.4.3-specs-2.patch
```

O primeiro patch desabilita o script **fixincludes** do GCC. Isto foi mencionado rapidamente antes, mas uma explicação mais detalhada do processo fixincludes agora é mais adequada. Sob circunstâncias normais, o script **fixincludes** do GCC faz uma varredura do sistema procurando cabeçalhos de arquivos que precisam ser reparados. Ele pode encontrar alguns cabeçalhos dos arquivos do Glibc do sistema anfitrião que necessitam de reparos; ele irá repará-los e movê-los para o diretório de inclusão privativo do GCC. No Chapter 6, depois que a nova Glibc for instalada, este diretório de inclusão privativo será rastreado antes do diretório de inclusão do sistema. O resultado é o GCC vai encontrar os cabeçalhos com reparos do sistema anfitrião, que muito provavelmente não combinarão com a versão da Glibc usada em nosso sistema LFS.

O segundo patch modifica a posição do vinculador dinâmico do GCC (tipicamente `ld-linux.so.2`). Também remove o diretório `/usr/include` do caminho de busca por inclusões do GCC. Aplicar os patches agora é melhor que modificar os arquivos de especificações depois da instalação do novo vinculador dinâmico durante a configuração do GCC. Isto é, todos os binários finais (e provisórios) criados durante esta configuração se ligarão com a nova Glibc.

Important

Estes patches são críticos e asseguram uma compilação bem sucedida. Não se esqueça de aplicá-los.

Crie um diretório separado de configuração outra vez:

```
mkdir ../gcc-build
cd ../gcc-build
```

Antes de começar a configurar o GCC, lembre-se de remover todas as variáveis de ambiente que modifiquem as opções de otimização padrão.

Prepare agora o GCC para a compilação:

```
../gcc-3.4.3/configure --prefix=/tools \
    --libexecdir=/tools/lib --with-local-prefix=/tools \
    --enable-clocale=gnu --enable-shared \
    --enable-threads=posix --enable-__cxa_atexit \
    --enable-languages=c,c++ --disable-libstdcxx-pch
```

Descrição das opções de configuração:

--enable-clocale=gnu
: Assegura que o modelo correto do locale é selecionado para as bibliotecas de C++ sob todas as circunstâncias. Se o script de configuração encontrar o locale *de_DE* instalado, selecionará o modelo GNU correto para o locale. Entretanto, se o locale *de_DE* não estiver instalado, há o risco de configurar uma Application Binary Interface (ABI) # incompatíveis com as bibliotecas C++ porque o modelo genérico incorreto de locale pode ser selecionado.

--enable-threads=posix
: Permite a manipulação das exceções do C++ para os códigos multi-threaded.

--enable-__cxa_atexit
: Permite o uso do *__cxa_atexit*, ao invés de *atexit*, para registrar os destructors do C++ para locais estáticos e objetos globais. Esta opção é essencial para a manipulação inteiramente padrão (standards-compliant) dos destructors. Afeta também a C++ ABI, o que conseqüentemente resulta em bibliotecas compartilhadas do C++ e em programas de C++ transportáveis para outras distribuições de Linux.

--enable-languages=c,c++
: Assegura que os compiladores de C e de C++ estejam configurados.

--disable-libstdcxx-pch
: Não compile com os cabeçalhos pré-compilados (PCH) para `libstdc++`. Ocupa muito espaço e nós não temos nenhum uso para eles.

Compile o pacote:

```
make
```

Não há nenhuma necessidade de usar agora a opção *bootstrap* porque o compilador que está sendo usado

para compilar este GCC foi compilado com a mesma versão das fontes do GCC usadas mais cedo.

A compilação está agora completa. Como mencionado anteriormente, executar o conjunto de testes para as ferramentas provisórias compiladas neste capítulo não é necessário. Para executar mesmo assim os testes do GCC, use o seguinte comando:

```
make -k check
```

A opção *-k* é usada para forçar a execução completa do conjunto de testes, não parando na primeira falha. O conjunto de testes do GCC é muito detalhado e é quase garantido que vai gerar algumas falhas. Para ver um sumário dos resultados dos testes, execute:

```
../gcc-3.4.3/contrib/test_summary
```

Para ver somente os sumários, direcione a saída através do **grep -A7 Summ**.

Os resultados podem ser comparados com os mostrados em *http://www.linuxfromscratch.org/lfs/build-logs/6.1/*.

Algumas falhas inesperadas não podem ser evitadas. Os desenvolvedores do GCC estão geralmente cientes destas situações, mas não as resolveram ainda. A menos que os resultados dos testes sejam muito diferentes daqueles no URL acima, é seguro continuar.

Instale o pacote:

```
make install
```

Note

Neste momento é altamente recomendado repetir a verificação de sanidade que nós executamos mais cedo neste capítulo. Consulte a Section 5.7, “Ajustando as ferramentas provisórias,” e repita a compilação do teste. Se o resultado der errado, a razão mais provável é que a correção das especificações do GCC não foi feita corretamente..

Os detalhes deste pacote estão na Section 6.14.2, “Conteúdo do GCC.”

5.12. Binutils-2.15.94.0.2.2 - segunda passagem

O pacote Binutils contém um vinculador dinâmico, um assembler, e outras ferramentas para manipular arquivos-objeto.

Tempo de compilação aproximado: 1.5 SBU
Espaço em disco necessário: 114 MB

Requisitos de instalação: Bash, Bison, Coreutils, Diffutils, Flex, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed e Texinfo

5.12.1. Re-instalação do Binutils

Este pacote é conhecido por apresentar problemas quando suas opções de otimização padrão (incluindo as opções *-march* e *-mcpu*) são modificadas. Se alguma variável de ambiente que modifique estas opções de otimização tiver sido definida, tais como `CFLAGS` e `CXXFLAGS`, remova-as quando for compilar este pacote.

Crie um diretório separado de configuração outra vez:

```
mkdir ../binutils-build
cd ../binutils-build
```

Prepare o Binutils para a compilação:

```
../binutils-2.15.94.0.2.2/configure --prefix=/tools \
    --disable-nls --enable-shared --with-lib-path=/tools/lib
```

Descrição das opções de configuração:

--with-lib-path=/tools/lib
Especifica o caminho de busca das bibliotecas durante a compilação do Binutils como sendo `/tools/lib` que será passado para o vinculador. Isto impede que o vinculador procure nos diretórios de bibliotecas do sistema anfitrião.

Compile o pacote:

```
make
```

A compilação está agora completa. Executar o conjunto de testes não é necessário para as ferramentas provisórias neste capítulo. Para executar mesmo assim os testes do Binutils, use o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Prepare agora o vinculador para a fase de “re-ajuste” do próximo capítulo:

```
make -C ld clean
make -C ld LIB_PATH=/usr/lib:/lib
```

Warning

Não remova as fontes do Binutils e os diretórios de trabalho ainda. Estes diretórios serão necessários mais uma vez no capítulo seguinte da forma que estão.

Os detalhes deste estão situados na Section 6.13.2, “Conteúdo do Binutils.”

5.13. Gawk-3.1.4

O pacote Gawk contém programas para manipular arquivos de texto.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 16.4 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Sed

5.13.1. Instalação do Gawk

Prepare o Gawk para a compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **make check**.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.20.2, “Conteúdo do Gawk.”

5.14. Coreutils-5.2.1

O pacote de Coreutils contém utilitários que permitem ver e ajustar as características básicas do sistema.

Tempo de compilação aproximado: 0.9 SBU
Espaço em disco necessário: 53.3 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl e Sed

5.14.1. Instalação do Coreutils

Prepare o Coreutils para a compilação:

```
DEFAULT_POSIX2_VERSION=199209 ./configure --prefix=/tools
```

Este pacote tem uma peculiaridade quando compilado com as versões da Glibc anteriores à 2.3.2. Alguns utilitários do Coreutils (tais como **head**, **tail** e **sort**) não aceitarão sua sintaxe tradicional, uma sintaxe utilizada por aproximadamente 30 anos. Esta antiga sintaxe é preservada por questão de compatibilidade até que nos lugares onde ainda é utilizada possa ser atualizada. Esta compatibilidade é conseguida ajustando a variável de ambiente `DEFAULT_POSIX2_VERSION` como “199209” no comando acima. Se você não quiser que o Coreutils mantenha compatibilidade com a sintaxe tradicional, omita o ajuste da variável de ambiente `DEFAULT_POSIX2_VERSION`. É importante lembrar que fazer isto terá conseqüências, incluindo a necessidade modificar muitos pacotes que ainda usam a sintaxe antiga. Conseqüentemente, recomenda-se que as instruções acima sejam seguidas

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make RUN_EXPENSIVE_TESTS=yes check`**. O *`RUN_EXPENSIVE_TESTS=yes`* diz ao conjunto de testes para executar diversos testes adicionais que são considerados relativamente #caros# (nos que toca ao uso do poder e da memória do processador central) em algumas plataformas, mas geralmente não é um problema sob Linux.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.15.2, “Conteúdo do Coreutils.”

5.15. Bzip2-1.0.3

O pacote Bzip2 contém programas para compressão e descompressão de arquivos. Arquivos de texto comprimindo com o **bzip2** alcançam uma porcentagem muito melhor de compressão do que com o tradicional **gzip**.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 3.5 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc e Make

5.15.1. Instalação do Bzip2

O pacote Bzip2 não contém um script de configuração **configure**. Compile-o com:

```
make
```

Para testar, use: **`make test`**.

Instale o pacote:

```
make PREFIX=/tools install
```

Os detalhes deste pacote estão em Section 6.40.2, “Conteúdo do Bzip2.”

5.16. Gzip-1.3.5

O pacote Gzip contém programas para comprimir e descomprimir arquivos.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 2.2 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make e Sed

5.16.1. Instalação do Gzip

Prepare o Gzip para a compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Este pacote não vem com um conjunto de testes.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.46.2, “Conteúdo do Gzip.”

5.17. Diffutils-2.8.1

O pacote Diffutils contém programas que mostram as diferenças entre arquivos ou diretórios.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 5.6 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Sed

5.17.1. Instalação do Diffutils

Prepare o Diffutils para a compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Este pacote não vem com um conjunto de testes.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.41.2, “Conteúdo do Diffutils.”

5.18. Findutils-4.2.23

O pacote Findutils contém programas para encontrar arquivos. Estes programas são utilizados para fazer buscas recursivas através de uma árvore do diretório e para criar, manter, e fazer buscas em uma base de dados (mais rapidamente do que em uma busca recursiva, mas irreal se a base de dados não foi atualizada recentemente).

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 8.9 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Sed

5.18.1. Instalação do Findutils

Prepare o Findutils para a compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **make check**.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.19.2, “Conteúdo do Findutils.”

5.19. Make-3.80

O pacote make contém um programa para compilar pacotes.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 7.1 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep e Sed

5.19.1. Instalação do Make

Prepare o Make para a compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.49.2, “Conteúdo do Make.”

5.20. Grep-2.5.1a

O pacote Grep contém programas para procurar em arquivos. É usado para exibir linhas de um arquivo que satisfazem determinado padrão.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 4.5 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Make, Sed e Texinfo

5.20.1. Instalação do Grep

Prepare o Grep para compilação:

```
./configure --prefix=/tools \
    --disable-perl-regexp
```

Descrição das opções de configuração:

--disable-perl-regexp

Assegura que o programa **grep** não faça ligação com uma biblioteca Perl Compatible Regular Expression (PCRE) que pode estar instalada no sistema anfitrião mas que não estará disponível quando da execução do ambiente **chroot**.

Compile o pacote:

```
make
```

Para testar os resultados, use: **make check**.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.44.2, “Conteúdo do Grep.”

5.21. Sed-4.1.4

O pacote de Sed contem um editor de fluxo (stream editor).

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 8.4 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Texinfo

5.21.1. Instalação do Sed

Prepare o Sed para compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.28.2, “Conteúdo do Sed.”

5.22. Gettext-0.14.3

O pacote Gettext contém utilitários para a internacionalização e localização. Eles permitem que os programas sejam compilados com suporte à língua nativa (NLS, Native Language Support) habilitando a exibição de mensagens de saída na língua nativa do usuário.

Tempo de compilação aproximado: 0.5 SBU
Espaço em disco necessário: 63.0 MB

Requisitos de instalação: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make e Sed

5.22.1. Instalação do Gettext

Prepare o Gettext para a compilação:

```
./configure --prefix=/tools --disable-libasprintf \
    --without-csharp
```

Descrição das opções de configuração:

--disable-libasprintf
Esta opção diz ao Gettext para não compilar a biblioteca `asprintf`. Nada neste capítulo ou no seguinte requer esta biblioteca e o Gettext será reconfigurado mais tarde.

--without-csharp
Assegura que Gettext não terá suporte para o compilador C# que pode estar presente no sistema anfitrião mas não estará disponível no ambiente **chroot**.

Compile o pacote:

```
make
```

Para testar os resultados, use: **make check**. Este exame é demorado, ao redor 7 SBUs. O conjunto de testes do Gettext falha sob determinadas condições conhecidas do sistema anfitrião, por exemplo quando encontra um compilador Java no anfitrião. Um patch experimental para desabilitar o Java está disponível no *http://www.linuxfromscratch.org/patches/*.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.30.2, “Conteúdo do Gettext.”

5.23. Ncurses-5.4

O pacote Ncurses contém bibliotecas para manipulação de caracteres de tela independentes ao terminal, para a criação de painéis e menus.

Tempo de compilação aproximado: 0.7 SBU
Espaço em disco necessário: 27.5 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make e Sed

5.23.1. Instalação do Ncurses

Prepare o Ncurses para a compilação:

```
./configure --prefix=/tools --with-shared \
    --without-debug --without-ada --enable-overwrite
```

Descrição das opções de configuração:

--without-ada
Assegura que o Ncurses não tenha suporte para o compilador Ada que pode estar presente no sistema anfitrião mas não estará disponível no ambiente **chroot**.

--enable-overwrite
O Ncurses instalará seus cabeçalhos de arquivos em `/tools/include`, ao invés de `/tools/include/ncurses`, para garantir que outros pacotes possam encontrá-los.

Compile o pacote:

```
make
```

Este pacote não vem com um suite do teste.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.21.2, “Conteúdo do Ncurses.”

5.24. Patch-2.5.4

O pacote Patch contém um programa para modificar ou criar arquivos aplicando um arquivo “patch” (remendo) especialmente criado pelo programa **diff**.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 1.5 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make e Sed

5.24.1. Instalação do Patch

Prepare o Patch para a compilação:

```
CPPFLAGS=-D_GNU_SOURCE ./configure --prefix=/tools
```

A opção *-D_GNU_SOURCE* somente é necessária em uma plataforma PowerPC. Pode ser excluída em outras arquiteturas.

Compile o pacote:

```
make
```

Este pacote não vem com um suite do teste.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.51.2, “Conteúdo do Patch.”

5.25. Tar-1.15.1

O pacote Tar contém um programa de empacotamento de arquivos (archiving)

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 12.7 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Sed

5.25.1. Instalação do Tar

Prepare o Tar para compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **make check**.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.57.2, “Conteúdo do Tar.”

5.26. Texinfo-4.8

O pacote de Texinfo contém programas para a leitura, a escrita, e conversão de páginas info.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 14.7 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses e Sed

5.26.1. Instalação do Texinfo

Prepare o Texinfo para a compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.34.2, “Conteúdo do Texinfo.”

5.27. Bash-3.0

O pacote bash contém o shell Bourne-Again SHell.

Tempo de compilação aproximado: 1.2 SBU
Espaço em disco necessário: 20.7 MB

Requisitos de instalação: Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses e Sed.

5.27.1. Instalação do Bash

O bash tem um problema quando compilado com as versões mais novas da Glibc. Este patch repara o problema:

```
patch -Np1 -i ../bash-3.0-avoid_WCONTINUED-1.patch
```

Prepare o Bash para a compilação:

```
./configure --prefix=/tools --without-bash-malloc
```

Descrição das opções de configuração:

--without-bash-malloc
Esta opção desativa o uso da função de alocação de memória do Bash (malloc) que causa falhas de segmentação. Desligando esta opção, o Bash usará as funções malloc da Glibc que são mais estáveis..

Compile o pacote:

```
make
```

Para testar os resultados, use: **make tests**.

Instale o pacote:

```
make install
```

Crie um vínculo para os programas que usam o **sh** como shell:

```
ln -s bash /tools/bin/sh
```

Os detalhes deste pacote estão na Section 6.37.2, “Conteúdo do Bash.”

5.28. M4-1.4.3

O pacote M4 contem um processador macros.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 2.8 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl e Sed

5.28.1. Instalação de M4

Prepare o M4 para a compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.24.2, “Conteúdo do M4.”

5.29. Bison-2.0

O pacote Bison contém um gerador de analisadores (parser generator).

Tempo de compilação aproximado: 0.6 SBU
Espaço em disco necessário: 10.0 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make e Sed

5.29.1. Instalação do Bison

Prepare o Bison para a compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.25.2, “Conteúdo do Bison.”

5.30. Flex-2.5.31

O pacote Flex contém um utilitário para gerar programas que reconhecem padrões em texto.

Tempo de compilação aproximado: 0.6 SBU
Espaço em disco necessário: 22.5 MB

Requisitos de instalação: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make e Sed

5.30.1. Instalação do Flex

O Flex contem diversos erros conhecidos, que podem ser reparados com o seguinte patch:

```
patch -Np1 -i ../flex-2.5.31-debian_fixes-3.patch
```

As ferramentas automáticas da GNU detectarão que o código fonte do Flex foi modificado pelo patch e tentará fazer a atualização da página do manual. Isto não funciona em muitos sistemas, e a página padrão é muito boa, assim certifique-se de que não seja modificada:

```
touch doc/flex.1
```

Prepare agora o Flex para a compilação:

```
./configure --prefix=/tools
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **make check**.

Instale o pacote:

```
make install
```

Os detalhes deste pacote estão em Section 6.29.2, “Conteúdo do Flex.”

5.31. Util-linux-2.12q

O pacote Util-linux contém programas diversos. Os mais importantes são usados para montar, desmontar, formatar, particionar e gerenciar discos rígidos, abrir portas tty e capturar mensagens do kernel.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 8.9 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed e Zlib

5.31.1. Instalação do Util-linux

A configuração padrão do Util-linux não usa os cabeçalhos e as bibliotecas recentemente instaladas em `/tools`. Isto é resolvido alterando o seu script de configuração:

```
sed -i 's@/usr/include@/tools/include@g' configure
```

Prepare o Util-linux para a compilação:

```
./configure
```

Compile algumas rotinas de suporte:

```
make -C lib
```

Somente alguns dos utilitários contidos neste pacote são necessários durante a configuração:

```
make -C mount mount umount
make -C text-utils more
```

Este pacote não vem com uma suite de testes.

Copie estes programas para o diretório de ferramentas provisório:

```
cp mount/{,u}mount text-utils/more /tools/bin
```

Os detalhes deste pacote estão em Section 6.59.3, “Conteúdo do Util-linux.”

5.32. Perl-5.8.6

O pacote do Perl contém a "Practical Extraction and Report Language".

Tempo de compilação aproximado: 0.8 SBU
Espaço em disco necessário: 79.8 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make e Sed

5.32.1. Instalação do Perl

Adapte antes alguns caminhos incorporados para a biblioteca C aplicando o seguinte patch:

```
patch -Np1 -i ../perl-5.8.6-libc-1.patch
```

Prepare o Perl para a compilação (Tenha certeza de que a expressão 'IO Fcntl POSIX' está corretamente grafada—todos os caracteres são letras, nenhum é número):

```
./configure.gnu --prefix=/tools -Dstatic_ext='IO Fcntl POSIX'
```

Descrição das opções de configuração:

-Dstatic_ext='IO Fcntl POSIX'
Diz o Perl para construir o jogo mínimo das extensões estáticas necessárias para a instalação e teste do pacote Coreutils no capítulo seguinte.

Somente alguns dos utilitários contidas neste pacote precisam ser configurados:

```
make perl utilities
```

Embora o Perl venha com um conjunto de testes, não se recomenda executá-lo neste momento. Somente uma parte do Perl foi compilada e executando **make test** agora vai fazer com que o resto do Perl seja compilado também, o que é desnecessário a esta altura. O conjunto de testes poderá ser executado no próximo capítulo.

Instale estas ferramentas e suas bibliotecas:

```
cp perl pod/pod2man /tools/bin
mkdir -p /tools/lib/perl5/5.8.6
cp -R lib/* /tools/lib/perl5/5.8.6
```

Os detalhes deste estão em Section 6.33.2, “Conteúdo do Perl.”

5.33. Stripping

As etapas desta seção são opcionais, mas se a partição LFS for muito pequena, é bom que os artigos desnecessários possam ser removidos. Os executáveis e as bibliotecas configuradas contém cerca de 130 MB de símbolos de depuração (debugging symbols) desnecessários. Remova estes símbolos com:

```
strip --strip-debug /tools/lib/*
strip --strip-unneeded /tools/{,s}bin/*
```

O segundo comando saltará cerca de vinte arquivos, relatando que não reconhece seu formato. A maioria deles são scripts e não binários.

Cuidado para *não* usar `--strip-unneeded` nas bibliotecas. As bibliotecas estáticas seriam destruídas e os pacotes do nosso jogo de ferramentas (toolchain) teriam que ser configurados outra vez.

Para recuperar outros 30 MB, remova a documentação:

```
rm -rf /tools/{info,man}
```

Temos agora pelo mais de 850 Mb de espaço livre extra no sistema de arquivos do LFS que pode ser usado para configurar e instalar a Glibc na fase seguinte. Se você puder configurar e instalar a Glibc, você poderá configurar e instalar todo o resto.

Part III. Configurando o sistema LFS

Chapter 6. Instalando o software do sistema básico

6.1. Introdução

Neste capítulo nós vamos entrar em nosso ambiente de trabalho e vamos finalmente iniciar a configuração do sistema LFS à sério. Isto é, nós vamos usar o chroot para entrar no mini sistema Linux provisório, vamos fazer algumas preparações finais e vamos então começar a instalar os pacotes.

A instalação destes softwares é precisa. Embora em muitos casos as instruções de instalação pudessem ser mais curtas e genéricas, nós optamos por fornecer as instruções completas para cada pacote a fim de minimizar as possibilidades de erros. A chave para aprender o que faz um sistema Linux em funcionamento é saber o que cada pacote faz e porque o usuário (ou o sistema) necessita dele. Para cada pacote instalado, um sumário de seu conteúdo é fornecido, seguido por breves descrições de cada programa e biblioteca do pacote instalado.

Para usar as opções de otimização do compilador durante este capítulo, reveja por favor as sugestões de otimização em *http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt*. As opções de otimização do compilador podem fazer um programa funcionar ligeiramente mais rápido, mas podem também causar dificuldades na compilação e problemas ao executar o programa. Se um pacote não compilar quando estiver sendo usada alguma opção de otimização, tente compilar sem este recurso e veja se isto elimina o problema. Mesmo quando um pacote compila usando alguma otimização, há o risco de ele ter sido compilado incorretamente por causa das interações complexas entre o código e as ferramentas de configuração. Os pequenos ganhos potenciais conseguidos com as opções de otimização do compilador freqüentemente não compensam os riscos envolvidos. Em sua primeira configuração de um sistema LFS recomendamos fazer todas as configurações sem qualquer otimização. Fique tranqüilo, o sistema resultante será bem rápido e estável ao mesmo tempo.

A ordem em que os pacotes são instalados neste capítulo deve ser seguida estritamente para termos certeza de que nenhum programa acidentalmente venha a incorporar nele como caminho de consulta o diretório `/tools`. Pela mesma razão, não compile pacotes em paralelo. Compilar em paralelo pode economizar tempo (em especial em máquinas com duas CPUs), mas pode resultar em um programa com o caminho incorporado para `/tools`, o que fará com que o programa não possa ser executado quando esse diretório for removido.

Antes das instruções de instalação, cada página da instalação fornece informações sobre o pacote, incluindo uma descrição concisa do seu conteúdo, o tempo aproximado da configuração, quanto espaço de disco é requerido durante o processo de configuração, e qualquer outra informação necessária para configurar com sucesso o pacote. Depois das instruções de instalação, há uma lista dos programas e das bibliotecas (junto com suas descrições breves) que o pacote instala.

Para acompanhar quais pacotes instalam quais arquivos em particular, um gerenciador de pacotes pode ser utilizado. Para uma visão geral de diversos estilos de gerenciadores de pacotes, consulte por favor a *http://www.linuxfromscratch.org/blfs/view/svn/introduction/important.html*. Para um método de gerenciamento de pacote especialmente adequado ao sistema LFS, nós recomendamos a leitura de *http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt*.

Note

Para o restante deste livro você deve fazer o login como usuário *root* e não mais como usuário *lfs*. Também tenha certeza de que a variável `$LFS` está definida.

6.2. Montando os sistemas de arquivos virtuais do kernel

Os vários sistemas de arquivos implementados diretamente pelo kernel são usados para comunicação do e para o próprio kernel. Estes sistemas de arquivos são virtuais, tanto que nenhum espaço em disco é utilizado por eles. O conteúdo destes sistemas de arquivo reside na memória.

Comece criando os diretórios onde os sistemas de arquivos serão montados:

```
mkdir -p $LFS/{proc,sys}
```

Monte agora os sistemas de arquivo:

```
mount -t proc proc $LFS/proc
mount -t sysfs sysfs $LFS/sys
```

Lembre-se que, se por qualquer razão você parar de trabalhar no sistema LFS para recomeçar mais tarde, é muito importante montar estes sistemas de arquivos novamente antes de usar o chroot.

Os sistemas de arquivo adicionais serão montados sob o ambiente chroot. Para manter o anfitrião atualizado, execute um “fake mount” para cada uma destes agora:

```
mount -f -t tmpfs tmpfs $LFS/dev
mount -f -t tmpfs tmpfs $LFS/dev/shm
mount -f -t devpts -o gid=4,mode=620 devpts $LFS/dev/pts
```

6.3. Entrando no ambiente Chroot

É hora de entrar no ambiente chroot para começar configurar e instalar o sistema LFS final. Como usuário *root*, execute o seguinte comando para entrar no reino que, neste momento, está habitado somente com as ferramentas provisórias:

```
chroot "$LFS" /tools/bin/env -i \
    HOME=/root TERM="$TERM" PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin:/tools/bin \
    /tools/bin/bash --login +h
```

A opção *-i*, dada ao comando **env**, cancelará todas as variáveis do ambiente chroot. Depois disso, apenas as variáveis HOME, TERM, PS1 e PATH são definidas de novo. A instrução *TERM=$TERM* vai definir a variável TERM dentro do chroot com o mesmo valor do ambiente externo ao chroot. Esta variável é necessária para programas como o **vim** e o **less** funcionarem corretamente. Se outras variáveis forem necessárias, tais como CFLAGS ou CXXFLAGS, este é um bom momento para defini-las.

A partir deste ponto não há mais nenhuma necessidade em usarmos a variável LFS, porque todo o trabalho estará restrito ao sistema de arquivos LFS. Isto acontece porque o shell do bash assume que $LFS é agora o diretório raiz (/).

Observe que /tools/bin vem por último no PATH. Isto significa que uma ferramenta provisória não será mais utilizada assim que sua versão final estiver instalada. Isto ocorre quando o shell não pode “se lembrar” das posições dos arquivos binários anteriormente executados — por esta razão a possibilidade de hashing foi desligada pelo parâmetro *+h* passado ao **bash**.

É importante que todos os comandos referidos no restante deste capítulo e também dos capítulos seguintes sejam executados dentro do ambiente chroot. Se você deixar este ambiente por qualquer razão (reinicializar o sistema, por exemplo), lembre-se de montar os sistemas de arquivo proc e devpts (como visto na seção anterior) e entrar no chroot outra vez antes de continuar com as instalações.

Note que o **bash** exibe a mensagem I have no name! isto á normal porque o arquivo /etc/passwd não foi criado ainda.

6.4. Mudança na propriedade

Neste momento, o proprietário do diretório `/tools` é o usuário *lfs*, um usuário que existe somente no sistema anfitrião. Embora o diretório `/tools` possa ser apagado quando o sistema LFS estiver terminado, pode ser conveniente mantê-lo para auxiliar na configuração de pacotes adicionais ao LFS. Se o diretório `/tools` for mantido como está, os arquivos são de propriedade de um user ID sem a correspondente conta. Isto é perigoso porque uma conta de usuário criada mais tarde poderia adotar o mesmo user ID, assumindo a propriedade do diretório `/tools` e de todos os arquivos nele existentes, expondo assim estes arquivos à manipulação mal-intencionada.

Para evitar que isso aconteça, acrescente o usuário *lfs* ao novo sistema LFS mais tarde, quando criarmos o arquivo `/etc/passwd`, tomando o cuidado de atribuir-lhe os mesmos user ID e group ID dados no sistema anfitrião. Outra alternativa é atribuir o conteúdo do diretório `/tools` ao usuário *root* com o seguinte comando:

```
chown -R 0:0 /tools
```

O comando utiliza *0:0* ao invés de *root:root*, porque o comando **chown** é incapaz de resolver o nome “root” até que o arquivo de senhas seja criado. Este livro supõe que você executou o comando **chown** desta forma.

6.5. Criando diretórios

É hora de criar uma estrutura de diretórios no sistema de arquivos do LFS. Crie uma árvore padrão de diretórios com os seguintes comandos:

```
install -d /{bin,boot,dev,etc/opt,home,lib,mnt}
install -d /{sbin,srv,usr/local,var,opt}
install -d /root -m 0750
install -d /tmp /var/tmp -m 1777
install -d /media/{floppy,cdrom}
install -d /usr/{bin,include,lib,sbin,share,src}
ln -s share/{man,doc,info} /usr
install -d /usr/share/{doc,info,locale,man}
install -d /usr/share/{misc,terminfo,zoneinfo}
install -d /usr/share/man/man{1,2,3,4,5,6,7,8}
install -d /usr/local/{bin,etc,include,lib,sbin,share,src}
ln -s share/{man,doc,info} /usr/local
install -d /usr/local/share/{doc,info,locale,man}
install -d /usr/local/share/{misc,terminfo,zoneinfo}
install -d /usr/local/share/man/man{1,2,3,4,5,6,7,8}
install -d /var/{lock,log,mail,run,spool}
install -d /var/{opt,cache,lib/{misc,locate},local}
install -d /opt/{bin,doc,include,info}
install -d /opt/{lib,man/man{1,2,3,4,5,6,7,8}}
```

Os diretórios, por padrão, são criados com as permissões no modo 755, mas isto não é desejável para todos os diretórios. Nos comandos acima, duas mudanças são feitas — uma no diretório home do usuário *root* e outra nos diretórios para arquivos temporários.

A primeira mudança nas permissões impede que qualquer um entre no diretório `/root` — da mesma forma que um usuário normal faria com seu próprio diretório home. A segunda mudança nas permissões garante que qualquer usuário possa escrever nos diretórios `/tmp` e `/var/tmp` mas não permite que apague de lá arquivos de outros usuários. Esta última proibição é definida pelo assim chamado “sticky bit,” o maior bit (1) na máscara 1777.

6.5.1. Nota sobre a adoção do padrão FHS

A árvore de diretório adotada é baseada no Filesystem Hierarchy Standard (FHS) (disponível em *http://www.pathname.com/fhs/*). Além da árvore criada por nós, este padrão prevê a existência dos diretórios `/usr/local/games` e `/usr/share/games`. O FHS não é preciso quando a estrutura de subdiretórios do `/usr/local/share`, então, nós criamos somente os diretórios que são necessários. Entretanto, sinta-se livre para criar estes diretórios se você preferir adotar com rigor o padrão FHS.

6.6. Criando vínculos simbólicos essenciais

Alguns programas têm incorporados caminhos para programas que não existem ainda em nosso sistema. Para satisfazer estes programas, crie alguns vínculos simbólicos que serão substituídos por arquivos reais no curso deste capítulo, depois que o software respectivo for instalado.

```
ln -s /tools/bin/{bash,cat,pwd,stty} /bin
ln -s /tools/bin/perl /usr/bin
ln -s /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -s bash /bin/sh
```

6.7. Criando os arquivos passwd, group, e de log

Para que o usuário *root* possa fazer o login e para que o nome “root” seja reconhecido, não podem faltar algumas entradas específicas nos arquivos `/etc/passwd` e `/etc/group`.

Críe o arquivo `/etc/passwd` com o seguinte comando:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
EOF
```

A senha real para o *root* (o “x” é usado aqui apenas para segurar o lugar) será definida mais tarde.

Crie o arquivo `/etc/group` com o seguinte comando:

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
EOF
```

Os grupos criados não são parte de nenhum padrão em especial — a decisão pela criação destes grupos se deve em parte pelas exigências de configuração do Udev, na seção seguinte, e em parte por ser uma convenção comum adotada por um grande número de distribuições Linux. A Linux Standard Base (LSB, disponível em *http://www.linuxbase.org*) recomenda que, além do do grupo “root” com um Group ID (GID) igual a 0, somente o grupo “bin” com um GID de 1 esteja presente. Todos os demais nomes de grupo e GIDs podem ser escolhidos livremente pelo administrador do sistema, uma vez que os programas bem-escritos não dependem dos números de GID, mas usam preferencialmente o nome do grupo.

Para remover o “I have no name!” do prompt, inicie um novo shell. Como a Glibc completa foi instalada no Chapter 5 e os arquivos `/etc/passwd` e `/etc/group` foram criados, a definição dos nomes de usuário e dos nomes de grupo é possível agora.

```
exec /tools/bin/bash --login +h
```

Note o uso do parâmetro *+h*. Isto diz ao **bash** para não usar o hashing. Sem esta opção, o **bash** recordaria o caminho dos binários executados. Desta forma, para assegurar o acesso aos binários compilados no curso deste capítulo tão logo sejam instalados, a opção *+h* será usada durante todo este capítulo.

Os programas **login**, **agetty** e **init** (entre outros) utilizam alguns arquivos de log para gravar informações do instante em que alguém faz o login no sistema em diante. Entretanto, estes programas não farão os registros nos arquivos de log se eles não existirem. Crie os arquivos de log e defina as permissões apropriadas:

```
touch /var/run/utmp /var/log/{btmp,lastlog,wtmp}
chgrp utmp /var/run/utmp /var/log/lastlog
chmod 664 /var/run/utmp /var/log/lastlog
```

O arquivo `/var/run/utmp` registra os usuários que estão logados no momento. O arquivo `/var/log/wtmp` registra todos os logins e logouts. O arquivo `/var/log/lastlog` registra quando cada usuário fez o login pela última vez. O arquivo `/var/log/btmp` registra as tentativas frustradas de login.

6.8. Ocupando o /dev

6.8.1. Criando nós iniciais dos dispositivos

Quando o kernel carrega o sistema, ele requer a presença de alguns vínculos com alguns dispositivos do sistema, em particular os dispositivos `console` e `null`. Crie-os com os seguintes comandos:

```
mknod -m 600 /dev/console c 5 1
mknod -m 666 /dev/null c 1 3
```

6.8.2. Montando o tmpfs e ocupando o /dev

O método recomendado de ocupação do diretório `/dev` com dispositivos é montar um sistema de arquivos virtual (tal como `tmpfs`) no diretório `/dev` e permitir que os dispositivos sejam criados dinamicamente nesse sistema de arquivos virtual enquanto são destacados ou alcançados. Isto geralmente é feito durante o processo de inicialização. Como este novo sistema não foi inicializado, é necessário fazer o que o pacote LFS-Bootscripts faria de outra maneira, montando o `/dev`:

```
mount -n -t tmpfs none /dev
```

O pacote de Udev é quem cria realmente os dispositivos no diretório `/dev`. Como ele será instalado somente mais tarde, no processo de configuração, crie manualmente um conjunto mínimo de ligações com dispositivos necessários para terminar o processo de configuração deste sistema:

```
mknod -m 622 /dev/console c 5 1
mknod -m 666 /dev/null c 1 3
mknod -m 666 /dev/zero c 1 5
mknod -m 666 /dev/ptmx c 5 2
mknod -m 666 /dev/tty c 5 0
mknod -m 444 /dev/random c 1 8
mknod -m 444 /dev/urandom c 1 9
chown root:tty /dev/{console,ptmx,tty}
```

Existem alguns vínculos e diretórios exigidos pelo LFS que são criados durante a inicialização do sistema pelo pacote LFS-Bootscripts. Como estamos em um ambiente chroot e não em um ambiente inicializado, estes vínculos e diretórios precisam ser criados agora:

```
ln -s /proc/self/fd /dev/fd
ln -s /proc/self/fd/0 /dev/stdin
ln -s /proc/self/fd/1 /dev/stdout
ln -s /proc/self/fd/2 /dev/stderr
ln -s /proc/kcore /dev/core
mkdir /dev/pts
mkdir /dev/shm
```

Finalmente, monte os sistemas de arquivos virtuais apropriados (do kernel) nos diretórios recém-criados:

```
mount -t devpts -o gid=4,mode=620 none /dev/pts
mount -t tmpfs none /dev/shm
```

A execução destes comandos **mount** pode resultar na seguinte mensagem:

```
can't open /etc/fstab: No such file or directory.
```

Este arquivo — `/etc/fstab` — não foi criado ainda, mas também não é necessário para que os sistemas de arquivos sejam montados corretamente. Assim, os avisos podem ser ignorados.

6.9. Linux-Libc-Headers-2.6.11.2

O pacote Linux-Libc-Headers contém os cabeçalhos do kernel “organizados”.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 26.9 MB

Requisitos de instalação: Coreutils

6.9.1. Instalação do Linux-Libc-Headers

Por anos, foi prática comum usar os cabeçalhos do kernel, em seu estado “natural” (direto do pacote do kernel), no diretório `/usr/include`, mas nos últimos anos, os desenvolvedores do kernel se colocaram firmemente contra isso. Isto fez surgir o projeto Linux-Libc-Headers, que foi projetado para manter uma versão estável da Application Programming Interface (API), dos cabeçalhos do kernel Linux.

Instale os arquivos dos cabeçalhos:

```
cp -R include/asm-i386 /usr/include/asm
cp -R include/linux /usr/include
```

Assegure-se de que todos os cabeçalhos sejam de propriedade do root:

```
chown -R root:root /usr/include/{asm,linux}
```

Certifique-se de que os usuários possam ler os cabeçalhos:

```
find /usr/include/{asm,linux} -type d -exec chmod 755 {} \;
find /usr/include/{asm,linux} -type f -exec chmod 644 {} \;
```

6.9.2. Conteúdo do Linux-Libc-Headers

Cabeçalhos instalados: /usr/include/{asm,linux}/*.h

Descrição rápida

`/usr/include/{asm,linux}/*.h` As APIs dos cabeçalhos do Linux

6.10. Man-pages-2.01

O pacote man-pages contém mais de 1.200 páginas do man.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 25.8 MB

Requisitos de instalação: Bash, Coreutils e Make

6.10.1. Instalação das Man-pages

Instale o pacote man-pages com o comando:

```
make install
```

6.10.2. Conteúdo do Man-pages

Arquivos instalados: várias páginas do man

Descrição rápida

`man pages`	Fornece descrição das funções C e C++, dos comandos e arquivos importantes do sistema, e dos arquivos de configuração mais importantes

6.11. Glibc-2.3.4

O pacote de Glibc contém a biblioteca C principal. Esta biblioteca fornece as rotinas básicas de alocação de memória, busca em diretórios, abertura e fechamento de arquivos, leitura e escrita de arquivos, manipulação de strings, "pattern matching", aritmética e assim por diante.

Tempo de compilação aproximado: 12.3 SBU
Espaço em disco necessário: 476

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Make, Perl, Sed e Texinfo

6.11.1. Instalação do Glibc

Este pacote é conhecido por apresentar problemas quando suas opções de otimização padrão (incluindo as opções *-march* e *-mcpu*) são modificadas. Se alguma variável de ambiente que modifique estas opções de otimização tiver sido definida, tais como `CFLAGS` e `CXXFLAGS`, remova-as quando for compilar este pacote.

O sistema de configuração do Glibc é independente e será instalado perfeitamente, mesmo que os arquivos de especificações do compilador e o vinculador dinâmico ainda estejam apontando para o diretório `/tools`. Estas especificações e o vinculador não podem ser ajustados antes que o Glibc esteja instalado porque os testes de autoconf do Glibc dariam resultados falsos e prejudicaria nosso objetivo de fazer uma compilação limpa.

O pacote linuxthreads contém as páginas do man para as bibliotecas instaladas pelo Glibc. Descompacta o pacote dentro do diretório de códigos-fonte do Glibc:

```
tar -xjvf /sources/glibc-linuxthreads-2.3.4.tar.bz2
```

O Glibc tem dois testes que falham quando o kernel em execução é o 2.6.11.x. Constatou-se que o problema estava relacionado com os próprios testes e não com o libc nem com o kernel. Este patch elimina o problema:

```
patch -Np1 -i ../glibc-2.3.4-fix_test-1.patch
```

A documentação do GCC recomenda que a sua configuração seja feita em um diretório dedicado de trabalho que não o diretório dos arquivos-fonte:

```
mkdir ../glibc-build
cd ../glibc-build
```

Prepare o Glibc para a compilação:

```
../glibc-2.3.4/configure --prefix=/usr \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.0 --libexecdir=/usr/lib/glibc
```

Descrição das novas opções de configuração:

--libexecdir=/usr/lib/glibc
: Modifica a posição do programa **pt_chown** de seu diretório padrão `/usr/libexec` para `/usr/lib/glibc`.

Compile o pacote:

```
make
```

Important

Nesta seção, a suite de testes da Glibc é considerado crítico. Não deixe de executá-los em nenhuma circunstância.

Teste o resultado:

```
make check
```

A suite de testes da Glibc é altamente dependente de determinadas funções do sistema anfitrião, em especial do kernel. Em geral, é de se esperar passar pela suite de testes do Glibc. Entretanto, em determinadas circunstâncias, algumas falhas são inevitáveis. Esta é uma lista das ocorrências mais comuns:

- Os testes com o *math* falham, as vezes, em sistemas onde o processador central não seja um Intel genuíno relativamente novo ou AMD genuíno. Algumas opções de otimização concorrem para isso também.
- Os testes com o *gettext* falham às vezes devido às características do sistema anfitrião. As razões para isso ainda não foram esclarecidas.
- Se você montou a partição LFS com a opção *`noatime`*, o teste *atime* vai falhar. Como mencionado em Section 2.4, “Montando a nova partição”, não use a opção *`noatime`* ao configurar o LFS.
- Quando executados em equipamentos mais antigos e lentos, alguns testes podem falhar por causa dos test timeouts sendo excedidos.

Embora seja uma mensagem inofensiva, a instalação da Glibc vai se queixar da ausência do arquivo `/etc/ld.so.conf`. Evite este aviso com o seguinte comando:

```
touch /etc/ld.so.conf
```

Instale o pacote:

```
make install
```

Os locales, que fazem o sistema responder na língua nativa do usuário, não foram instalados pelo comando acima. Instale-os com:

```
make localedata/install-locales
```

Para ganhar tempo, uma alternativa ao comando precedente (que gera e instala o locale para cada biblioteca da Glibc) é instalar somente aqueles locales que você quer ou necessita. Isso pode ser feito usando o comando **localedef**. As informações deste comando podem ser encontradas no arquivo `INSTALL`, nos arquivos-fonte da Glibc. Entretanto, alguns locales são essenciais para os testes de alguns pacotes a serem instalados, em especial os testes *libstdc++* do GCC. As seguintes instruções, ao invés da *`install-locales`* usado acima, vão instalar um conjunto mínimo dos locales necessários para os testes funcionarem com sucesso:

```
mkdir -p /usr/lib/locale
localedef -i de_DE -f ISO-8859-1 de_DE
```

```
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i ja_JP -f EUC-JP ja_JP
```

Alguns locales instalados pelo comando **make localedata/install-locales** acima não são compatíveis com algumas aplicações que estão nos livros do LFS e do BLFS. A razão de vários problemas surge devido aos programadores das aplicações adotarem estruturas que não suportam tais locales, o LFS não deve ser usado com os locales que utilizam multibyte character sets (incluindo o UTF-8) ou a ordem de escrita da-direita-para-a-esquerda. Muitos patches não oficiais e instáveis são necessários para corrigir estes problemas, e foi decidido pelos colaboradores do projeto LFS não dar suporte a locales muito complexos. Isto aplica-se aos locales do ja_JP e do fa_IR — eles são instalados somente para passar nos testes do GCC e do Gettext, e o programa **watch** (parte do pacote Procps) não irá funcionar apropriadamente neles. As várias tentativas de contornar estas limitações estão documentadas nas sugestões referentes à internacionalização.

Compile as páginas man dos linuxthreads, que são uma grande referência sobre a threading API (aplicável ao NPTL também):

```
make -C ../glibc-2.3.4/linuxthreads/man
```

Instale estas páginas:

```
make -C ../glibc-2.3.4/linuxthreads/man install
```

6.11.2. Configurando a Glibc

O arquivo `/etc/nsswitch.conf` precisa ser criado porque, embora a Glibc siga determinados padrões quando este arquivo falta, ou está corrompido, os padrões da Glibc não trabalham bem assim em um ambiente de rede. A zona de tempo também precisa ser configurada.

Crie um novo arquivo `/etc/nsswitch.conf` com o seguinte comando:

```
cat > /etc/nsswitch.conf << "EOF"
# Begin /etc/nsswitch.conf

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

protocols: files
services: files
ethers: files
rpc: files
```

```
# End /etc/nsswitch.conf
EOF
```

Para determinar fuso horário local, execute o seguinte script:

```
tzselect
```

Após responder algumas perguntas sobre sua localização, o script gera uma saída com o nome correspondente ao seu fuso horário (e.g., *EST5EDT* ou *Canada/Eastern*). Crie então o arquivo `/etc/localtime` com o comando:

```
cp --remove-destination /usr/share/zoneinfo/[xxx] \
    /etc/localtime
```

Substitua *`[xxx]`* pelo nome do fuso horário fornecido pelo **tzselect** (por exemplo, Canada/Eastern).

O significado dos parâmetros usados com o cp:

`--remove-destination`
: Isto é necessário para forçar a remoção da vinculação simbólica já existente. A razão para copiar o arquivo em vez de usar uma vinculação simbólica é porque o diretório `/usr` está em outra partição. Isto será importante quando se fizer o login na modalidade único usuário.

6.11.3. Configurando o Vinculador Dinâmico

Por padrão, o vinculador dinâmico (`/lib/ld-linux.so.2`) procura em `/lib` e `/usr/lib` pelas bibliotecas dinâmicas necessárias aos programas em execução. Entretanto, se houver bibliotecas em outros diretórios que não o `/lib` e o `/usr/lib`, eles precisam ser acrescentados ao arquivo `/etc/ld.so.conf` para que o vinculador dinâmico possa localizá-las. Dois diretórios que geralmente têm bibliotecas adicionais são o `/usr/local/lib` e o `/opt/lib`, então acrescente estes diretórios ao caminho de busca do vinculador dinâmico.

Críe um novo arquivo `/etc/ld.so.conf` com o comando:

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf

/usr/local/lib
/opt/lib

# End /etc/ld.so.conf
EOF
```

6.11.4. Conteúdo do Glibc

Programas instalados: catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, nscd_nischeck, pcprofiledump, pt_chown, rpcgen, rpcinfo, sln, sprof, tzselect, xtrace, zdump e zic
Bibliotecas instaladas: ld.so, libBrokenLocale.[a,so], libSegFault.so, libanl.[a,so], libbsd-compat.a, libc.[a,so], libcrypt.[a,so], libdl.[a,so], libg.a, libieee.a, libm.[a,so], libmcheck.a, libmemusage.so, libnsl.a,

libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libnss_nis.so, libnss_nisplus.so, libpcprofile.so, libpthread.[a,so], libresolv.[a,so], librpcsvc.a, librt.[a,so], libthread_db.so e libutil.[a,so]

Descrição rápida

catchsegv	É usado para obter informações da pilha quando um programa termina com falha de segmentação
gencat	Gera catálogos de mensagens
getconf	Exibe os valores da configuração do sistema de variáveis específicas do sistema de arquivos
getent	Obtém entradas de uma base de dados administrativa
iconv	Faz conversões de conjunto de caracteres
iconvconfig	Cria um módulo de configuração de rápido acesso para o **iconv**
ldconfig	Configura as ligações de tempo de execução do vinculador dinâmico
ldd	Exibe as bibliotecas compartilhadas requeridas por cada programa ou biblioteca especificada na linha de comando
lddlibc4	Auxilia o**ldd** na manipulação de arquivos-objetos
locale	É um programa Perl que diz ao compilador para habilitar (ou desabilitar) o uso de localidades POSIX para operações embutidas (built-in)
localedef	Compila especificações de localidade
mtrace	Lê e interpreta um arquivo de trace de memória e mostra um sumário em formato legível por um humano
nscd	É um daemon que provê um cache para os pedidos de serviços de nomes mais comuns
nscd_nischeck	Verifica se o modo seguro é ou não necessário para pesquisas NIS+
pcprofiledump	Exibe informação gerada por perfis PC
pt_chown	Configura o dono, grupo e permissão de acesso do pseudo-terminal escravo correspondente ao pseudo-terminal mestre fornecido pelo descritor de arquivo '3'. Este é o programa auxiliar da função **grantpt**. Ele não é destinado à execução direta a partir da linha de comando.
rpcgen	Gera código C para a implementação do protocolo RPC
rpcinfo	Faz uma chamada RPC para um servidor RPC
sln	Liga simbolicamente o destino à origem. Ele é estaticamente vinculado, não precisando de vinculação dinâmica. Por isso, o **ln**é útil para fazer vinculações simbólicas para bibliotecas dinâmicas se o sistema dinâmico de ligação não estiver funcional por algum motivo.
sprof	Lê e exibe dados de perfis de objetos compartilhados
tzselect	Pede informações ao usuário sobre a sua localização atual e exibe na saída padrão a descrição do fuso horário resultante
xtrace	Investiga a execução de um programa exibindo a função atualmente executada

zdump	Exibe o fuso horário
zic	É o compilador do fuso horário
`ld.so`	É o programa auxiliar para executáveis de bibliotecas compartilhadas
`libBrokenLocale`	Usados por programas, como o Mozilla, para a correção de locales defeituosos
`libSegFault`	Gerenciador de sinais de falha de segmentação
`libanl`	Biblioteca assíncrona de pesquisa de nomes
`libbsd-compat`	Provê a portabilidade necessária para executar certos programas da distribuição de Berkey (DEB) no Linux
`libc`	A biblioteca C principal
`libcrypt`	A biblioteca de criptografia
`libdl`	A biblioteca de interface para a vinculação dinâmica
`libg`	Biblioteca em tempo de execução para o **g++**
`libieee`	Biblioteca de ponto flutuante IEEE
`libm`	Biblioteca matemática
`libmcheck`	Contém código executado na inicialização
`libmemusage`	É utilizado pelo **memusage** para ajudar a coletar informação sobre uso da memória por um programa
`libnsl`	A biblioteca de serviços de rede
`libnss`	A biblioteca Name Service Switch, contendo funções para gerenciamento de nomes do servidores, de usuários, de grupos, de apelidos, de serviços, de protocolos etc.
`libpcprofile`	Código usado pelo kernel para calcular o tempo de CPU gasto em funções, linhas de código-fonte e instruções
`libpthread`	A biblioteca POSIX de processos
`libresolv`	Contém funções destas bibliotecas permitem a criação, envio e interpretação de pacotes de servidores DNS da Internet
`librpcsvc`	Contém funções para diversos serviços RPC
`librt`	Contém funções com interfaces especificadas pela Extensão em Tempo Real POSIX.1b
`libthread_db`	Contém funções úteis para a construção de depuradores para programas multiprocessados
`libutil`	Contém funções “padronizadas” usadas em diferentes utilitários Unix

6.12. Re-ajustando as ferramentas provisórias

Agora que as bibliotecas C definitivas estão instaladas, é o momento de ajustar outra vez as ferramentas provisórias. Elas serão ajustadas de modo fazer a vinculação dos programas recém-compilados com as novas bibliotecas. É o mesmo processo usado na fase "Ajustando as ferramentas provisórias" do começo do Chapter 5 mas agora com os ajustes invertidos. No Chapter 5, o movimento guiada no sentido dos diretórios `/{,usr/}lib` do sistema anfitrião para o novo diretório `/tools/lib`. Agora, o movimento será deste mesmo diretório `/tools/lib` para os diretórios `/{,usr/}lib`.

Comece ajustando o vinculador dinâmico (linker). Os diretórios dos arquivos fonte e de configuração da segunda passagem do Binutils foram mantidos com esta finalidade. Instale o vinculador dinâmico devidamente ajustado executando o seguinte comando dentro do diretório `binutils-build`:

```
make -C ld INSTALL=/tools/bin/install install
```

Note

Se a advertência anterior para manter os arquivos fonte e o diretório de configuração do Binutils da segunda passagem no Chapter 5 não foi observada, ou se acidentalmente forem apagados ou estão inacessíveis, ignore o comando acima. O resultado será que o próximo pacote, Binutils, se ligará com as bibliotecas C em `/tools` ao invés das em `/{,usr/}lib`. Isto não é o ideal, entretanto os testes vão mostrar que os programas binários resultantes do Binutils devem ser os mesmos.

De agora em diante, cada programa compilado será vinculado somente às bibliotecas em `/usr/lib` e `/lib`. A opção *`INSTALL=/tools/bin/install`* é necessária porque o arquivo `Makefile` criado durante a segunda passagem contém ainda referências à **/usr/bin/install** que ainda não foi instalado. Algumas distribuições do anfitrião têm uma ligação simbólica com o `ginstall` que faz exame de precedência no arquivo `Makefile` pode causar problemas. O comando acima toma previne isso.

Remova os arquivos de fonte e o diretório de configuração do Binutils agora.

Em seguida, modifique o arquivo de especificações do GCC para que aponte para o novo linker dinâmico. Um comando **perl** fará isto:

```
perl -pi -e 's@ /tools/lib/ld-linux.so.2@ /lib/ld-linux.so.2@g;' \
    -e 's@\*startfile_prefix_spec:\n@$_/usr/lib/ @g;' \
        `gcc --print-file specs`
```

É uma boa idéia inspecionar visualmente o arquivo de especificações para verificar se a mudança pretendida foi realmente efetuada.

Important

Se estiver trabalhando em uma plataforma onde o vinculador dinâmico não seja `ld-linux.so.2`, então substitua "ld-linux.so.2" pelo nome do vinculador dinâmico da sua plataforma nos comandos acima. Consulte o Section 5.2, "Notas técnicas sobre as ferramentas provisórias," se necessário.

Caution

É importante neste momento fazer uma pausa para assegurar-se de que as funções básicas (compilação e vinculação) das ferramentas provisórias estão funcionando como esperado. Para fazer isto, execute uma verificação de sanidade:

```
echo 'main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /lib'
```

Se tudo estiver funcionando corretamente, não deve haver nenhum erro, e a saída do último comando será (permitindo diferenças específicas de cada plataforma no nome do vinculador dinâmico):

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

Note que agora o prefixo de nosso vinculador dinâmico é `/lib`.

Se a saída não for como esta mostrada acima nem não houver nenhuma saída, então algo está seriamente errado. Investigue e repasse todas as etapas para encontrar onde o problema está para corrigi-lo. A razão mais provável é que algo saiu errado com a modificação no arquivo de especificações do GCC. Todas as pendências precisam ser resolvidas antes de continuar com o processo de configuração.

Uma vez que tudo esteja funcionando corretamente, apague os arquivos dos teste:

```
rm dummy.c a.out
```

6.13. Binutils-2.15.94.0.2.2

O pacote Binutils contém um vinculador dinâmico, um assembler, e outras ferramentas para manipular arquivos-objeto.

Tempo de compilação aproximado: 1.3 SBU
Espaço em disco necessário: 158 MB

Requisitos de instalação: Bash, Bison, Coreutils, Diffutils, Flex, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed e Texinfo

6.13.1. Instalação do Binutils

Este pacote é conhecido por apresentar problemas quando suas opções de otimização padrão (incluindo as opções *-march* e *-mcpu*) são modificadas. Se alguma variável de ambiente que modifique estas opções de otimização tiver sido definida, tais como `CFLAGS` e `CXXFLAGS`, remova-as quando for compilar este pacote.

Certifique-se de que os PTYs estão trabalhando corretamente dentro do ambiente chroot. Verifique se tudo está bem ajustado com um teste muito simples:

```
expect -c "spawn ls"
```

Se a seguinte mensagem aparecer, o ambiente do chroot não está corretamente ajustado para operação com PTY:

```
The system has no more ptys.
Ask your system administrator to create more.
```

Esta questão precisa ser resolvida antes de executar o conjunto de testes do Binutils e do GCC.

A documentação deste pacote recomenda que sua configuração seja realizada em um diretório de trabalho diferente do diretório dos arquivos fonte:

```
mkdir ../binutils-build
cd ../binutils-build
```

Prepare o Binutils para a compilação:

```
../binutils-2.15.94.0.2.2/configure --prefix=/usr \
    --enable-shared
```

Compile o pacote:

```
make tooldir=/usr
```

Normalmente, o tooldir (o diretório onde os executáveis serão finalmente colocados) é ajustado para `$(exec_prefix)/$(target_alias)`. Por exemplo, máquinas i686 convertem isto para `/usr/i686-pc-linux-gnu`. Como estamos configurando um sistema customizado, este diretório-alvo, em `/usr`, não é necessário. O `$(exec_prefix)/$(target_alias)` seria usado se o sistema fosse usado para compilação cruzada (por exemplo, compilando um pacote em uma máquina Intel gerando código que poderia ser executado em máquinas PowerPC).

Important

O conjunto de testes para o Binutils nesta seção é considerado crítico. Não pule esta etapa em hipótese alguma.

Teste os resultados:

```
make check
```

Instale o pacote:

```
make tooldir=/usr install
```

Instale o arquivo de cabeçalhos do `libiberty` que são necessários para alguns pacotes:

```
cp ../binutils-2.15.94.0.2.2/include/libiberty.h /usr/include
```

6.13.2. Conteúdo do Binutils

Programas instalados: addr2line, ar, as, c++filt, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings e strip
Bibliotecas instaladas: libiberty.a, libbfd.[a,so] e libopcodes.[a,so]

Descrição rápida

addr2line Converte os endereços de um programa em nomes de arquivo e números de linhas. Dado um endereço e um executável, ele usa a informação de depuração do executável para identificar qual nome de arquivo e número de linha estão associados ao endereço dado

ar Cria arquivos-conteiner, modifica e extrai de arquivos de arquivos-conteiner (archives). Um arquivo-conteiner contém uma coleção de outros arquivos em uma estrutura que torna possível recuperar os arquivos individuais originais (chamados membros do arquivo)

as Assembler que converte a saída do compilador GCC em arquivos-objeto (utilizados pelo vinculador ld)

c++filt Usado pelo linker para repassar símbolos C++ e Java e para livrar funções sobrecarregadas por conflitos

gprof Exibe dados do perfil de gráficos de chamada

ld Um vinculador que concatena vários arquivos-objetos, organiza seu conteúdo e vincula referências simbólicas. Freqüentemente, a última etapa na compilação de um novo programa é feita pelo ld

nm Lista os símbolos de arquivos-objetos

objcopy Copia o conteúdo de um arquivo-objeto para outro. O objcopy utiliza a biblioteca GNU BFD para ler e escrever os arquivos-objetos. Ele pode escrever o arquivo final em um formato diferente do arquivo original

objdump Exibe informações sobre um ou mais arquivos-objetos. As opções controlam qual informação

particular será exibida. Este programa é útil para programadores que estão produzindo ferramentas de compilação

ranlib Gera um índice do conteúdo de um arquivo e armazena esta informação no próprio arquivo. O índice lista cada símbolo definido pelos arquivos-objetos vinculáveis pertencentes ao arquivo

readelf Exibe informações sobre binários do tipo ELF

size Lista os tamanhos das seções (e o tamanho total) para cada um dos arquivos-objetos fornecidos. Por padrão, uma linha de informação é gerada para cada arquivo-objeto ou módulo em um arquivo

strings Para cada arquivo fornecido, strings exibe as seqüências de caracteres imprimíveis com pelo menos 4 caracteres de comprimento (ou o número especificado em uma das opções do programa) seguidas por um caractere não-imprimível. Por padrão, ele apenas exibe as strings das seções inicializadas e carregadas dos arquivos-objetos. Para outros tipos de arquivo, ele imprime as strings do arquivo todo. O strings é principalmente útil para determinar o conteúdo de arquivos que não estão em formato texto

strip Descarta todos ou determinados símbolos de arquivos-objetos. A lista de arquivos-objetos pode incluir arquivos contendo outros arquivos. Ao menos um arquivo-objeto deve ser fornecido. strip modifica os arquivos fornecidos, ao invés de fazer cópias com nomes diferentes dos arquivos modificados

`libiberty` Contém as rotinas usadas por vários programas GNU, incluindo **getopt**, **obstack**, **strerror**, **strtol** e **strtoul**

`libbfd` Biblioteca de Descrição de Arquivos Binários

`libopcodes` Uma biblioteca para lidar com códigos de operações de processador (opcodes)—gerando uma versão “legível” das instruções do processador; é usado para compilar utilitários tais como o **objdump**.

6.14. GCC-3.4.3

O pacote do GCC contém uma coleção de compiladores GNU, que inclui os compiladores C e C++.

Tempo de compilação aproximado: 11.7 SBU
Espaço em disco necessário: 451 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, Gettext, Glibc, Grep, Make, Perl, Sed e Texinfo

6.14.1. Instalação do GCC

Este pacote é conhecido por apresentar problemas quando suas opções de otimização padrão (incluindo as opções *-march* e *-mcpu*) são modificadas. Se alguma variável de ambiente que modifique estas opções de otimização tiver sido definida, tais como `CFLAGS` e `CXXFLAGS`, remova-as quando for compilar este pacote.

Aplique somente o patch No-Fixincludes patch (mas não o patch Specs) como fizemos no capítulo anterior:

```
patch -Np1 -i ../gcc-3.4.3-no_fixincludes-1.patch
```

O GCC falha ao compilar alguns pacotes que não são da instalação básica do Linux From Scratch (por exemplo, Mozilla e kdegraphics) quando usado em conjunto com as versões mais novas do Binutils. Use o seguinte patch para corrigir este problema:

```
patch -Np1 -i ../gcc-3.4.3-linkonce-1.patch
```

Faça uma substituição com o **sed** para suprimir a instalação do `libiberty.a`. A versão do `libiberty.a` fornecido pelo Binutils é que será usada:

```
sed -i 's/install_to_$(INSTALL_DEST) //' libiberty/Makefile.in
```

A documentação deste pacote recomenda que sua configuração seja realizada em um diretório de trabalho diferente do diretório dos arquivos fonte:

```
mkdir ../gcc-build
cd ../gcc-build
```

Prepare o GCC para a compilação:

```
../gcc-3.4.3/configure --prefix=/usr \
    --libexecdir=/usr/lib --enable-shared \
    --enable-threads=posix --enable-__cxa_atexit \
    --enable-clocale=gnu --enable-languages=c,c++
```

Compile o pacote:

```
make
```

Important

Nesta seção, o conjunto de testes para o GCC é considerado crítico. Não deixe de executá-lo sob nenhuma circunstância.

Teste os resultados, mas não pare nos erros:

```
make -k check
```

Alguns dos erros são ocorrências conhecidas e foram mencionados no capítulo precedente. Os comentários feitos para o conjunto de testes na Section 5.11, “GCC-3.4.3 - Pass 2,” são aplicáveis também aqui. Consulte-as se necessário.

Instale o pacote:

```
make install
```

Alguns pacotes esperam que o preprocessador C esteja instalado no diretório `/lib`. Para dar suporte àqueles pacotes, crie um vínculo:

```
ln -s ../usr/bin/cpp /lib
```

Muitos pacotes usam o nome **cc** para chamar o compilador de C. Para dar suporte àqueles pacotes, crie um vínculo:

```
ln -s gcc /usr/bin/cc
```

Note

Neste momento, recomenda-se fortemente repetir a verificação de sanidade executada mais cedo neste capítulo. Consulte Section 6.12, “Re-ajustando as ferramentas provisórias,” e repita a verificação. Se os resultados apontarem algum erro, a razão mais provável será que o patch Specs do GCC aplicado no Chapter 5 foi erroneamente aplicado aqui.

6.14.2. Conteúdo do GCC

Programas instalados: c++, cc (link to gcc), cpp, g++, gcc, gccbug e gcov
Bibliotecas instaladas: libgcc.a, libgcc_eh.a, libgcc_s.so, libstdc++.[a,so] e libsupc++.a

Descrição rápida

cc O compilador C

cpp O pré-processador C; é usado pelo compilador para processar as instruções #include, #define e outras similares nos arquivos fonte

c++ O compilador C++

g++ O compilador C++

gcc O compilador C

gccbug Um shell script muito útil usado para criar relatórios de erro

gcov Uma ferramenta de teste de otimização; é usado para indicar onde as opções de otimização terão melhores efeitos.

`libgcc` Suporte em tempo de execução para o **gcc**

`libstdc++` A biblioteca C++ padrão

`libsupc++` Rotinas de suporte à linguagem de programação C++

6.15. Coreutils-5.2.1

O pacote de Coreutils contém utilitários que permitem ver e ajustar as características básicas do sistema.

Tempo de compilação aproximado: 0.9 SBU
Espaço em disco necessário: 52.8 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl e Sed

6.15.1. Instalação de Coreutils

Um problema conhecido com o programa **uname** deste pacote é que a opção *-p* retorna sempre `unknown`. O seguinte patch corrige este comportamento para as máquinas tipo Intel:

```
patch -Np1 -i ../coreutils-5.2.1-uname-2.patch
```

Impeça que Coreutils instale binários que serão instalados por outros pacotes mais tarde:

```
patch -Np1 -i \
    ../coreutils-5.2.1-suppress_uptime_kill_su-1.patch
```

Prepare agora Coreutils para a compilação:

```
DEFAULT_POSIX2_VERSION=199209 ./configure --prefix=/usr
```

Compile o pacote:

```
make
```

O conjunto de testes do Coreutils faz diversas suposições sobre a presença de usuários e de grupos que são inválidos no ambiente mínimo que temos neste momento. Conseqüentemente, ítens adicionais precisam ser criados antes da execução dos testes. Salte para “Instale o pacote” se você não pretende executar o conjunto de testes.

Crie dois grupos dummy e um usuário dummy:

```
echo "dummy1:x:1000:" >> /etc/group
echo "dummy2:x:1001:dummy" >> /etc/group
echo "dummy:x:1000:1000:::/bin/bash" >> /etc/passwd
```

Agora o conjunto de testes está pronto para ser executado. Primeiro, execute os testes, que devem ser executados como usuário *root*:

```
make NON_ROOT_USERNAME=dummy check-root
```

Execute então os demais testes como usuário *dummy*:

```
src/su dummy -c "make RUN_EXPENSIVE_TESTS=yes check"
```

Quando o teste estiver completo, remova o usuário e os grupos dummy:

```
sed -i '/dummy/d' /etc/passwd /etc/group
```

Instale o pacote:

```
make install
```

Mova programas para os locais apropriadas:

```
mv /usr/bin/{[,basename,cat,chgrp,chmod,chown,cp,dd,df} /bin
mv /usr/bin/{date,echo,false,head,hostname,install,ln} /bin
mv /usr/bin/{ls,mkdir,mknod,mv,pwd,rm,rmdir,sync} /bin
mv /usr/bin/{sleep,stty,test,touch,true,uname} /bin
mv /usr/bin/chroot /usr/sbin
```

Finalmente, crie um vínculo simbólico para ser FHS-compatível:

```
ln -s ../../bin/install /usr/bin
```

6.15.2. Conteúdo do Coreutils

Programas instalados: basename, cat, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, hostname, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mv, nice, nl, nohup, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, rm, rmdir, seq, sha1sum, shred, sleep, sort, split, stat, stty, sum, sync, tac, tail, tee, test, touch, tr, true, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami e yes

Descrição rápida

basename Remove diretórios e sufixos de um nome de arquivo fornecido

cat Concatena arquivos para a saída padrão

chgrp Muda a propriedade do grupo de arquivos e diretórios

chmod Muda as permissões de cada arquivo para a modalidade especificada; a modalidade pode ser uma representação simbólica das mudanças a fazer ou um número octal que representa as novas permissões

chown Muda a propriedade do usuário e/ou do grupo de arquivos e dos diretórios

chroot Executa um comando ou uma nova instância do shell com um diretório especificado funcionando como diretório `/`

cksum Exibe a assinatura CRC e contagem de bytes de cada arquivo especificado

comm Compara linha por linha dois arquivos ordenados

cp Copia arquivos

csplit Exibe partes de um arquivo separadas por um padrão em arquivos xx01, xx02, ... e retorna a contagem de bytes de cada parte

cut Exibe partes selecionadas de linhas dos arquivos especificados para a saída padrão

date Exibe a data e horário atuais em um formato especificado, ou configura a data do sistema

dd Copia um arquivo (da entrada padrão para a saída padrão, por padrão) usando tamanhos de blocos de entrada e saída especificados, enquanto está fazendo opcionalmente conversões nele

df Relata a quantidade de espaço de disco disponível (e usado) em todos os sistemas de lima montada, ou somente nos sistemas de lima que prendem as limas selecionadas

dir Lista o conteúdo do diretório especificado (igual ao comando **ls**)

dircolors Fornece uma seqüência comandos para ajustar a variável de ambiente de LS_COLOR que define as cores desejadas para os comandos **ls**

dirname Remove sufixos que não são diretórios de um nome de arquivo

du Relata a quantidade de espaço de disco usada pelo diretório atual, por cada um dos diretórios dados (todos os subdiretórios incluídos) ou por cada uma das limas dadas

echo Exibe uma linha de texto

env Executa um comando em um ambiente modificado

expand Converte tabulações para espaços em arquivos, mostrando na saída padrão

expr Avalia expressões

factor Exibe os fatores primos de todos os números inteiros especificados

false Não faz nada, e falha; sempre termina com um código de status indicando erro

fmt Reformata os parágrafos dos arquivos especificados, escrevendo para a saída padrão

fold Quebra linhas de cada arquivo especificado (ou entrada padrão), escrevendo para a saída padrão

groups Exibe os grupos dos quais um usuário faz parte

head Imprime na saída padrão as primeiras dez linhas (ou o número de linhas especificado) de um arquivo

hostid Exibe o identificador numérico (em hexadecimal) do sistema

hostname Relata ou ajusta o nome do anfitrião

id Exibe os IDs de usuário e de grupo do usuário atual ou de algum outro especificado

install Copia arquivos enquanto define suas permissões e, se possível, seus proprietário e grupo

join Une linhas de dois arquivos em um campo comum

link Cria um hardlynk para um arquivo com o nome fornecido

ln Cria um hardlynk ou um vínculo simbólico (soft link) entre arquivos

logname Exibe nome de login do usuário atual

ls Lista o conteúdo de um diretório

md5sum Mostra ou verifica assinaturas MD5

mkdir Cria diretórios

mkfifo Cria um First-In, First-Outs (FIFOs), um “named pipe” no jargão UNIX

mknod Cria #device nodes#; um #device nodes# é um arquivo de caracteres especiais, blocos especiais, ou um FIFO

mv Move ou renomeia arquivos e diretórios

nice Executa um programa com prioridade modificada

nl Numera na saída padrão as linhas dos arquivos especificados

nohup Executa um comando imune a interrupções, com saída para um arquivo de log

od Mostra uma representação não ambígua, em octal por padrão, de um arquivo especificado

paste Funde arquivos, escrevendo para a saída padrão linhas consistindo em linhas seqüenciais de cada arquivo especificado, separado por TABs

pathchk Verifica se nomes de arquivo são válidos ou independentes do sistema (portáveis)

pinky Um finger simples que obtém informações sobre um certo usuário

pr Formata em páginas ou colunas para impressão

printenv Exibe todas ou algumas das variáveis de ambiente

printf Formata e exibe dados (da mesma forma que a função printf do C)

ptx Produz um índice permutado do conteúdo dos arquivos especificados, com cada palavra-chave em seu contexto

pwd Exibe o nome do diretório atual

readlink Relata o valor de uma ligação simbólica

rm Remove arquivos e diretórios

rmdir Remove os diretórios se estiverem vazios

seq Exibe números de uma certa faixa com um determinado incremento

sha1sum Imprime ou verifica somas de controle seguras do algoritmo 1 da mistura 160-bit (SHA1)

shred Remoção segura de arquivos. Sobrescreve a posição do arquivo no disco repetidamente com testes de padrões complexos, tornando impossível a recuperação de dados

sleep Cria uma espera no sistema durante determinado tempo

sort Classifica as linhas dos arquivos especificados

split Divide um arquivo partes, pelo tamanho ou pelo número de linhas

stat Exibe o status de um arquivo ou de u sistema de arquivos

stty Modifica e exibe as configurações da linha do terminal

sum Exibe a assinatura e contagem de blocos para cada arquivo especificado

sync Esvazia os buffers do sistema de arquivos; isto força a gravação dos blocos alterados no disco

tac Concatena para a saída padrão arquivos na ordem inversa

tail Imprime as últimas dez linhas (ou o número de linhas informado) de cada lima especificado

tee Lê da entrada padrão para escrever na saída padrão e em um arquivo especificado

test	Compara valores e verifica tipos de arquivos
touch	As mudanças arquivam timestamps, ajustando os tempos do acesso e da modificação das limas dadas ao tempo atual; as limas que não existem são criadas com o comprimento zero
tr	Traduz, comprime e/ou remove caracteres da entrada padrão, escrevendo para a saída padrão
true	Não faz nada, mas com sucesso; sempre termina com um código de status indicando sucesso
tsort	Escreve listas totalmente ordenadas consistentes com a ordenação parcial dos arquivos especificados
tty	Exibe o nome de arquivo do terminal conectado à entrada padrão
uname	Exibe informações sobre o sistema
unexpand	Converte espaços de cada arquivo para TABs, escrevendo para a saída padrão
uniq	Mostra linhas duplicadas de um arquivo
unlink	Chama a função unlink para remover um arquivo
users	Exibe os nomes dos usuários atualmente logados no sistema
vdir	É o mesmo que **ls -l**
wc	Exibe a contagem de linhas, palavras e bytes para cada arquivo especificado e uma linha total, se mais de um arquivo foi especificado
who	Mostra quem está logado
whoami	Mostra o nome de usuário associado ao UID atual
yes	Gera uma saída "y" ou uma string dada repetidamente, até ser terminado

6.16. Zlib-1.2.2

O pacote Zlib contem rotinas de compressão e descompressão usadas por alguns programas.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 2.7 MB

Requisitos de instalação: Binutils, Coreutils, GCC, Glibc, Make e Sed

6.16.1. Instalação do Zlib

A Zlib tem uma vulnerabilidade de estouro de buffer (buffer overflow). O seguinte patch resolve o problema:

```
patch -Np1 -i ../zlib-1.2.2-security_fix-1.patch
```

Note

A Zlib configura incorretamente sua biblioteca compartilhada quando a variável `CFLAGS` está definida no ambiente. Se a variável `CFLAGS` estiver definida, acrescente a opção *`-fPIC`* à variável `CFLAGS` durante a execução do comando de configuração abaixo, removendo-a depois de completado o processo.

Prepare o Zlib para a compilação:

```
./configure --prefix=/usr --shared --libdir=/lib
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale a biblioteca compartilhada:

```
make install
```

O comando anterior instalou um arquivo `.so` em `/lib`. Vamos removê-lo e revinculá-lo à `/usr/lib`:

```
rm /lib/libz.so
ln -sf ../../lib/libz.so.1.2.2 /usr/lib/libz.so
```

Configure a biblioteca estática:

```
make clean
./configure --prefix=/usr
make
```

Para testar outra vez os resultados, use: **`make check`**.

Instale a biblioteca estática:

```
make install
```

Modifique as permissões de acesso da biblioteca estática:

```
chmod 644 /usr/lib/libz.a
```

6.16.2. Conteúdo do Zlib

Bibliotecas instaladas: libz.[a,so]

Descrição rápida

`libz` Contém as funções de compressão e descompressão usadas por alguns programas

6.17. Mktemp-1.5

O pacote de Mktemp contém os programas usados para criar arquivos temporários seguros em scripts para o shell.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 436 KB

Requisitos de instalação: Coreutils, Make e Patch

6.17.1. Instalação do Mktemp

Muitos /scripts/ usam ainda o depreciado programa **tempfile**, que tem função similar ao **mktemp**. Aplique este patch ao Mktemp para incluir um wrapper para o **tempfile**:

```
patch -Np1 -i ../mktemp-1.5-add_tempfile-2.patch
```

Prepare o Mktemp para a compilação:

```
./configure --prefix=/usr --with-libc
```

Descrição das opções de configuração:

--with-libc
Faz o programa **mktemp** usar as funções *mkstemp* e *mkdtemp* da biblioteca C do sistema.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
make install-tempfile
```

6.17.2. Conteúdo do Mktemp

Programas instalados: mktemp e tempfile

Descrição rápida

mktemp	Cria arquivos temporários de maneira segura; é usada em scripts.
tempfile	Cria arquivos temporários de maneira menos segura do que o **mktemp**; é instalado por questão de compatibilidade

6.18. Iana-Etc-1.04

O pacote Iana-Etc fornece dados para os serviços de rede e protocolos.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 1.9 MB

Requisitos de instalação: Make

6.18.1. Instalação do Iana-Etc

O seguinte comando converte os dados em estado bruto fornecidos pelo IANA para os formatos corretos para arquivos de dados dos diretórios `/etc/protocols` e `/etc/services`:

```
make
```

Instale o pacote:

```
make install
```

6.18.2. Conteúdo do Iana-Etc

Arquivos instalados: /etc/protocols e /etc/services

Descrição rápida

`/etc/protocols`	Descreve os vários protocolos DARPA da Internet que estão disponíveis no subsistema TCP/IP
`/etc/services`	Traça um mapa de nomes textuais amigáveis para serviços da Internet, suas portas e protocolos

6.19. Findutils-4.2.23

O pacote Findutils contém programas para encontrar arquivos. Estes programas são utilizados para fazer buscas recursivas através de uma árvore do diretório e para criar, manter, e fazer buscas em uma base de dados (mais rapidamente do que em uma busca recursiva, mas irreal se a base de dados não foi atualizada recentemente).

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 9.4 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Sed

6.19.1. Instalação do Findutils

Prepare o Findutils para a compilação:

```
./configure --prefix=/usr --libexecdir=/usr/lib/locate \
    --localstatedir=/var/lib/locate
```

A opção *localstatedir* modifica a localização da base de dados do **locate** para `/var/lib/locate`, o que é compatível com o FHS.

Compile o pacote:

```
make
```

Para testar os resultados, use: **make check**.

Instale o pacote:

```
make install
```

6.19.2. Conteúdo do Findutils

Programas instalados: bigram, code, find, frcode, locate, updatedb e xargs

Descrição rápida

bigram Foi usado anteriormente gerar a bases de dados do **locate**

code Foi usado anteriormente gerar bases de dados do **locate**; é o antecessor do **frcode**.

find Buscas na árvore de diretórios por arquivos que combinam os critérios especificados

frcode É chamado pelo **updatedb** para comprimir a lista de nomes de arquivos; usa um método de compressão front-compression que que reduz o tamanho da base de dados por um fator de quatro a cinco.

locate Buscas através de uma base de dados por nomes de arquivos e apresenta relatório com os arquivos cujos nomes contém a string fornecida ou que combinam com o padrão especificado

updatedb Atualiza a base de dados do **locate**; faz a varredura do sistema de arquivos inteiro (outros sistemas de arquivos que estejam montados são incluídos também, a menos que dito que não) e põe cada nome de arquivo por ele encontrado, e sua localização, na base de dados

xargs Pode ser usado aplicar um comando a uma lista de arquivos

6.20. Gawk-3.1.4

O pacote Gawk contém programas para manipular arquivos de texto.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 16.4 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Sed

6.20.1. Instalação do Gawk

Prepare o Gawk para a compilação:

```
./configure --prefix=/usr --libexecdir=/usr/lib
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

6.20.2. Conteúdo do Gawk

Programas instalados: awk (link to gawk), gawk, gawk-3.1.4, grcat, igawk, pgawk, pgawk-3.1.4 e pwcat

Descrição rápida

awk	Um vínculo para o **gawk**
gawk	Um programa para manipular arquivos de texto; é a implementação GNU do **awk**
gawk-3.1.4	Um hardlynk para o **gawk**
grcat	Despeja a base de dados de grupos do `/etc/group`
igawk	Dá ao **gawk** a habilidade de incluir arquivos
pgawk	A versão profiling do **gawk**
pgawk-3.1.4	Um hardlynk para o **pgawk**
pwcat	Despeja a base de dados de senhas `/etc/passwd`

6.21. Ncurses-5.4

O pacote Ncurses contém bibliotecas para manipulação de caracteres de tela independentes ao terminal, para a criação de painéis e menus.

Tempo de compilação aproximado: 0.6 SBU
Espaço em disco necessário: 18.6 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make e Sed

6.21.1. Instalação do Ncurses

Prepare o Ncurses para a compilação:

```
./configure --prefix=/usr --with-shared --without-debug
```

Compile o pacote:

```
make
```

Este pacote não vem com um conjunto de testes.

Instale o pacote:

```
make install
```

Definir as permissões de execução das bibliotecas Ncurses:

```
chmod 755 /usr/lib/*.5.4
```

Corrigindo as permissões de uma biblioteca que não deve ser executável:

```
chmod 644 /usr/lib/libncurses++.a
```

Mova as bibliotecas para o diretório `/lib`, onde elas devem ficar:

```
mv /usr/lib/libncurses.so.5* /lib
```

Como as bibliotecas foram movidas, alguns vínculos apontam para arquivos não existentes. Recreie estresse vínculos simbólicos:

```
ln -sf ../../lib/libncurses.so.5 /usr/lib/libncurses.so
ln -sf libncurses.so /usr/lib/libcurses.so
```

6.21.2. Conteúdo do Ncurses

Programas instalados: captoinfo (link to tic), clear, infocmp, infotocap (link to tic), reset (link to tset), tack, tic, toe, tput e tset
Bibliotecas instaladas: libcurses.[a,so] (link to libncurses.[a,so]), libform.[a,so], libmenu.[a,so], libncurses++.a, libncurses.[a,so] e libpanel.[a,so]

Descrição rápida

captoinfo	Converte uma descrição do termcap em uma descrição do terminfo
clear	Limpa a tela, se isto for possível. Ele identifica no ambiente o tipo de terminal e varre a base de dados do terminfo para descobrir como limpar a tela
infocmp	Compara ou exibe descrições do terminfo. Pode ser usado para comparar uma entrada binária do terminfo com outras entradas, reescrever uma descrição para ter vantagem no uso do campo use, ou exibir uma descrição do arquivo binário (term) em uma variedade de formatos (o oposto do que o tic faz)
infotocap	Converte uma descrição do terminfo em uma descrição do termcap
reset	Reinicializa um terminal para seus valores padrão. Configura os modos cooked e echo, desativa os modos cbreak e raw, ativa a conversão de nova-linha e zera qualquer caractere especial não configurado para os seus valores-padrão antes de fazer a inicialização do terminal da mesma forma que tset.
tack	O verificador da ação do terminfo; é usada principalmente testar a exatidão de uma entrada na base de dados do terminfo
tic	É o compilador do terminfo para entradas de descrições. O programa converte um arquivo terminfo do formato fonte para o formato binário para uso com as rotinas da biblioteca ncurses. Arquivos terminfo contém informação sobre as capacidades de um terminal.
toe	Lista todos os tipos de terminais disponíveis, dando o nome e a descrição preliminar de cada um
tput	Usa a base de dados do terminfo para criar os valores das capacidades dependentes de terminal e informações disponíveis no shell, para inicializar ou reinicializar o terminal ou retornar o nome longo do tipo de terminal requisitado
tset	Pode ser usado para inicializar terminais
`libcurses`	um vínculo com `libncurses`
`libncurses`	Estas bibliotecas são a base do sistema e são usadas para exibir texto (freqüentemente de um modo requintado) na tela. Um exemplo onde o Ncurses é usado é no processo **make menuconfig** do kernel
`libform`	Contém funções para executar formulários
`libmenu`	Contém funções para executar menus
`libpanel`	Contém funções para executar os painéis

6.22. Readline-5.0

O pacote de Readline é um conjunto de bibliotecas que acrescenta recursos de edição e de histórico à linha de comando do shell.

Tempo de compilação aproximado: 0.11 SBU
Espaço em disco necessário: 9.1 MB

Requisitos de instalação: Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses e Sed

6.22.1. Instalação do Readline

O seguinte patch corrige um problema onde o Readline mostra às vezes somente 33 caracteres em uma linha e então quebra para a linha seguinte. Inclui também outros reparos recomendados pelo autor do Readline.

```
patch -Np1 -i ../readline-5.0-fixes-1.patch
```

Prepare o Readline para a compilação:

```
./configure --prefix=/usr --libdir=/lib
```

Compile o pacote:

```
make SHLIB_XLDFLAGS=-lncurses
```

O significado da opção do make:

SHLIB_XLDFLAGS=-lncurses
Esta opção força o Readline a fazer um vínculo com a biblioteca `libncurses`.

Instale o pacote:

```
make install
```

Dê permissões mais apropriadas para as bibliotecas dinâmicas do Readline:

```
chmod 755 /lib/lib{readline,history}.so*
```

Mova agora as bibliotecas estáticas para um lugar mais apropriado:

```
mv /lib/lib{readline,history}.a /usr/lib
```

Em seguida, remova as arquivos `.so` de `/lib` e revincule-os à `/usr/lib`.

```
rm /lib/lib{readline,history}.so
ln -sf ../../lib/libreadline.so.5 /usr/lib/libreadline.so
ln -sf ../../lib/libhistory.so.5 /usr/lib/libhistory.so
```

6.22.2. Conteúdo do Readline

Conteúdo do: libhistory.[a,so] e libreadline.[a,so]

Descrição rápida

`libhistory`	Fornece uma interface consistente com o usuário recordando o histórico da linha de comando
`libreadline`	Ajuda a dar consistência à interface com o de usuário através da linha do comando

6.23. Vim-6.3

O pacote Vim contém um editor de texto poderoso.

Tempo de compilação aproximado: 0.4 SBU
Espaço em disco necessário: 38.0 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses e Sed

Alternativas ao Vim

Se você preferir algum outro editor de texto —como o Emacs, Joe ou Nano—consulte *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html* para instruções de instalação.

6.23.1. Instalação do Vim

Primeiro, desempacote os arquivos `vim-6.3.tar.bz2` e (opcionalmente) `vim-6.3-lang.tar.gz` no mesmo diretório. Então, mude as os locais padrão dos arquivos de configuração `vimrc` e `gvimrc` para o diretório `/etc`:

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
echo '#define SYS_GVIMRC_FILE "/etc/gvimrc"' >> src/feature.h
```

O pacote Vim tem um vulnerabilidade na segurança já identificada. O seguinte patch elimina o problema:

```
patch -Np1 -i ../vim-6.3-security_fix-1.patch
```

Agora prepare o Vim para a compilação:

```
./configure --prefix=/usr --enable-multibyte
```

O opcional mas altamente recomendado parâmetro *`--enable-multibyte`* inclui suporte para edição de arquivos com caracteres codificados em mais de um byte (multibyte character encodings) ao **vim**. Isto é necessário quando usando um locale com a definição de caracteres multibyte. Este parâmetro é útil também para permitir a edição de arquivos de texto criados em distribuições Linux como a Fedora Core que utiliza a UTF-8 como conjunto de caracteres padrão.

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make test`**. Entretanto, este conjunto de testes gera saída com muitos dados binários na tela, que pode causar problemas com os ajustes atuais do terminal. Isto pode ser resolvido redirecionando a saída para um arquivo de log.

Instale o pacote:

```
make install
```

Muitos usuários costumam usar **vi** ao invés de **vim**. Para permitir a execução do **vim** quando os usuários usarem o comando **vi**, crie um vínculo:

```
ln -s vim /usr/bin/vi
```

Se um sistema de janelas X (X Window System) vier a ser instalado no sistema LFS, pode ser necessária a recompilação do Vim após a instalação do X. O Vim tem uma versão GUI do editor que requer o X e algumas bibliotecas adicionais para ser instalado. Para mais informação sobre este processo, consulte a documentação do Vim e a página da instalação Vim no livro do projeto BLFS em *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html#postlfs-editors-vim*.

6.23.2. Configurando o Vim

Por padrão, o **vim** é executado em um modo incompatível com o vi. Isto pode ser novidade para os usuários acostumados com editores antigos. A opção “nocompatible” é incluída abaixo para destacar o fato de que um novo modo está sendo usado. Lembra também àqueles que mudariam para o modo “compatible” que este deve ser o primeiro ajuste no arquivo de configuração. Isto é necessário porque muda outros ajustes, e se sobrepões ao que deve vir após este ajuste. Crie um arquivo de configuração padrão para o **vim**com o sequinte comando:

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

O *set nocompatible* faz o **vim** se comportar de um modo mais útil (o padrão) do que no modo vi-compatível. Remova o “no” para manter o velho modo **vi**. O *set backspace=2* permite retroceder sobre quebra de linhas, autoindentações e no começo da inserção. A linha *syntax on* habilita o destaque de sintaxe do vim. Finalmente, o operador *if* com a instrução *set background=dark* corrige a suposição do **vim**'s guess sobre a cor do fundo em alguns emuladores de terminal. Isto dá o destaque a um esquema de cores melhor para o uso no fundo preto destes programas.

A documentação para outras opções disponíveis pode ser obtida executando o seguinte comando:

```
vim -c ':options'
```

6.23.3. Conteúdo do Vim

Programas instalados: efm_filter.pl, efm_perl.pl, ex (link to vim), less.sh, mve.awk, pltags.pl, ref, rview (link to vim), rvim (link to vim), shtags.pl, tcltags, vi (link to vim), view (link to vim), vim, vim132, vim2html.pl, vimdiff (link to vim), vimm, vimspell.sh, vimtutor e xxd

Descrição rápida

efm_filter.pl Um filtro que lê da entrada padrão, copia para a saída padrão e cria um arquivo de erro que pode ser lido pelo **vim**

efm_perl.pl Reformata as mensagens de erro do interpretador Perl para o uso com o modo “quickfix” do **vim**

ex Inicia o **vim** em modo Ex

less.sh Um script que inicia o **vim** com less.vim

mve.awk Processa erros do**vim**

pltags.pl Cria um arquivo de tags (marcações) para códigos Perl a ser usado pelo **vim**

ref Verifica a gramática dos argumentos

rview Versão restrita do **view**; nenhum comando shell pode ser iniciado e o **view**não pode ser suspenso

rvim Versão restrita do **vim**; nenhum comando shell pode ser iniciado e o **vim** não pode ser suspenso

shtags.pl Gera um arquivo de tags para scripts Perl

tcltags Gera um arquivo de tags para código TCL

view Inicia o **vim** em modo somente leitura

vi É o editor

vim É o editor

vim132 Inicia o **vim** em modo de terminal com 132 colunas

vim2html.pl Converte a documentação do vim para HypterText Markup Language (HTML)

vimdiff Edita duas ou três versões de um arquivo com o **vim** e exibe as diferenças

vimm Habilita o modelo de entrada do localizador DEC em um terminal remoto

vimspell.sh Script que verifica a gramática de um arquivo e gera as instruções de sintaxe necessárias para o destaque no **vim**. Este script requer o velho comando **spell** do Unix, o qual não está disponível nem no LFS e nem no BLFS

vimtutor Ensina o básico sobre as teclas chave e os comandos do **vim**

xxd Exibe um arquivo em hexadecimal ou faz o inverso

6.24. M4-1.4.3

O pacote M4 contem um processador macros.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 2.8 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl e Sed

6.24.1. Instalação do M4

Configure o pacote para compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Para testar o resultado, use: **make check**.

Instale o pacote:

```
make install
```

6.24.2. Conteúdo do M4

Programas instalados: m4

Descrição rápida

m4 É um processador de macros. Ele copia a entrada para a saída, expandindo macros à medida em que aparecem. Essas macros são nativas ou definidas pelo usuário e podem receber qualquer número de argumentos. Além de fazer expansão de macros, o **m4** tem funções nativas para a inclusão de arquivos, execução de comandos Unix, cálculo de aritmética de inteiros, manipulação de texto de diversas formas, recursividade, etc. O programa **m4** pode ser usado como um front-end (interface) para um compilador ou como um processador de macros independente.

6.25. Bison-2.0

O pacote Bison contém um gerador de analisadores (parser generator).

Tempo de compilação aproximado: 0.6 SBU
Espaço em disco necessário: 9.9 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make e Sed

6.25.1. Instalação do Bison

Configure o pacote para compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Para testar os resultados, use: `make check`.

Instale o pacote:

```
make install
```

6.25.2. Conteúdo do Bison

Programas instalados: bison e yacc
Biblioteca instalada: liby.a

Descrição rápida

bison O bison é um gerador de analisadores (parser generator), um substituto para o yacc. Yacc significa Outro Compilador de Compiladores (Yet Another Compiler Compiler).

yacc Este script para o Bash executa o bison usando a opção **bison**, feito com propósitos de compatibilidade com programas que utilizam o **yacc** ao invés do **bison**; ele chama **bison** com o parâmetro *-y*

`liby.a` A biblioteca de Yacc que contém execuções das funções *yyerror* e *main*; esta biblioteca não é normalmente muito útil, mas o POSIX a exige

6.26. Less-382

O pacote Less contém um visualizador de arquivos de texto.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 2.3 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses e Sed

6.26.1. Instalação do Less

Prepare o Less para a compilação:

```
./configure --prefix=/usr --bindir=/bin --sysconfdir=/etc
```

Descrição das opções de configuração:

`--sysconfdir=/etc`
Esta opção diz aos programas criados pelo pacote para procurar em `/etc` pelos arquivos de configuração.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

6.26.2. Conteúdo do Less

Programas instalados: less, lessecho e lesskey

Descrição Rápida

less	Um visualizador ou paginador de arquivos de texto; exibe o conteúdo do arquivo especificado, com recursos de rolagem pelo usuário, busca por strings e salto para marcações
lessecho	Necessário para expandir meta-caracteres, tais como * e ?, nos nomes de arquivo em sistemas Unix
lesskey	Usado para especificar as teclas de atalho **less**

6.27. Groff-1.19.1

O pacote de Groff contém programas para processar e formatar texto. O Groff converte texto puro e comandos especiais para determinado formato, como o que você vê em uma página de manual.

Tempo de compilação aproximado: 0.5 SBU
Espaço em disco necessário: 38.7 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make e Sed

6.27.1. Instalação do Groff

O Groff faz a leitura da variável de ambiente `PAGE` que deve conter o tamanho padrão do papel utilizado para impressão. Para usuários nos Estados Unidos *`PAGE=letter`* é apropriado. Em outros lugares, *`PAGE=A4`* pode ser mais adequado.

Prepare o Groff para a compilação:

```
PAGE=[paper_size] ./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Alguns programas de documentação, tais como o **xman**, funcionarão corretamente sem os seguintes vínculos:

```
ln -s soelim /usr/bin/zsoelim
ln -s eqn /usr/bin/geqn
ln -s tbl /usr/bin/gtbl
```

6.27.2. Conteúdo do Groff

Programas instalados: addftinfo, afmtodit, eqn, eqn2graph, geqn (link to eqn), grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl (link to tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pfbtops, pic, pic2graph, post-grohtml, pre-grohtml, refer, soelim, tbl, tfmtodit, troff e zsoelim (vículo para o soelim)

Descrição rápida

addftinfo	Lê um arquivo de fontes troff e adiciona informações sobre tamanho de fontes que são usadas pelo sistema **groff**
afmtodit	Cría um arquivo de fontes para uso com o **groff** e o **grops**
eqn	Compila descrições de equações embutidas em arquivos de entrada troff em comandos que são entendidos pelo **troff**
eqn2graph	Converte um EQN (equação) do troff em uma imagem
geqn	Um vínculo para o **eqn**

grn Um pré-processador do **groff** para arquivos gremlin

grodvi Um driver para o **groff** que provê o formato dvi do TeX

groff Um front-end para o sistema de formatação de documentos groff. Normalmente ele executa o programa **troff** e um pós-processador apropriado para o dispositivo selecionado

groffer Exibe arquivos groff e páginas do man no X e em terminais tty

grog Lê arquivos e detecta quais das opções `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, e `-t`, do **groff**, são necessárias para imprimir cada arquivo exibindo o comando **groff** com as opções necessárias

grolbp É um driver do **groff** para as impressoras Canon CAPSL (impressoras a laiser das séries LBP-4 e LBP-8)

grolj4 É um driver do **groff** que produz saída no formato PCL5 apropriado para uma impressora HP Laserjet 4

grops Traduz a saída do GNU **troff** para PostScript

grotty Traduz a saída do GNU **troff** um formato apropriado para dispositivos tipo máquina de escrever

gtbl Um vínculo para o **tbl**

hpftodit Cria um arquivo de fontes para uso com o **groff -Tlj4** a partir de um arquivo de tamanho de fontes HP

indxbib Cria um índice remissivo para base de dados bibliográficos em um arquivo para uso com **refer**, **lookbib** e **lkbib**

lkbib Procura referências em bases de dados bibliográficos que contém palavras-chaves específicas e exibe as referências encontradas

lookbib Exibe uma linha de comando na saída de erros padrão (a não ser que a saída padrão não seja um terminal), lê da entrada padrão uma linha contendo um conjunto de caracteres, procura por referências contendo estes caracteres em bases de dados bibliográficos em um arquivo especificado, exibe na saída padrão as referências encontradas e repete este processo até o final do arquivo

mmroff Um pré-processador simples para o **groff**

neqn Formata equações para saída ASCII

nroff Um script emula o comando **nroff** usando o **groff**

pfbtops Traduz uma fonte Postscript no formato `.pfb` para o ASCII

pic Compila as descrições de figuras embutidas em arquivos de entrada troff ou TeX em comandos que são entendidos pelo TeX ou **troff**

pic2graph Converte um diagrama PIC em uma imagem

post-grohtml Traduz a saída do GNU **troff** para HTML

pre-grohtml Traduz a saída do GNU **troff** para HTML

refer o conteúdo de um arquivo para a saída padrão, exceto linhas entre *.[* e *.]*, interpretadas como citações, e linhas entre *.R1* e *.R2*, interpretadas como comandos que definem como as citações serão processadas

soelim Lê arquivos e substitui linhas na forma *arquivo .so* pelo conteúdo de arquivo *arquivo*

tbl Compila descrições de tabelas embutidas em arquivos de entrada troff em comandos que são entendidos pelo **troff**

tfmtodit Cria um arquivo de fontes para uso com **groff -Tdvi**

troff É altamente compatível com o Unix **troff**. Normalmente ele deve ser invocado usando o comando **groff**, que irá executar também pré e pós-processadores na ordem apropriada e com as opções apropriadas

zsoelim Um vínculo para o **soelim**

6.28. Sed-4.1.4

O pacote de Sed contem um editor de fluxo (stream editor).

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 8.4 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Texinfo

6.28.1. Instalação do Sed

Prepare o Sed para a compilação:

```
./configure --prefix=/usr --bindir=/bin
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

6.28.2. Conteúdo do Sed

Programa instalado: sed

Descrição Rápida

sed Filtra e transforma arquivos texto em uma única passagem.Um editor de stream é usado para fazer transformações básicas em textos de um stream de entrada (um arquivo ou entrada de um canal).

6.29. Flex-2.5.31

O pacote Flex contém um utilitário para gerar programas que reconhecem padrões em texto.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 22.5 MB

Requisitos de instalação: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make e Sed

6.29.1. Instalação do Flex

O Flex contém diversos erros conhecidos. Elimine-os com o seguinte:

```
patch -Np1 -i ../flex-2.5.31-debian_fixes-3.patch
```

As ferramentas automáticas (autotools) GNU detectam que o código fonte do Flex foi modificado pelo patch aplicado e tentam atualizar também sua página man. Isto não funciona corretamente em muitos sistemas e a página padrão é muito boa. Evite isso:

```
touch doc/flex.1
```

Prepare o Flex para a compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

Há alguns pacotes que esperam encontrar em a biblioteca `lex` em `/usr/lib`. Crie um vínculo para resolver isto:

```
ln -s libfl.a /usr/lib/libl.a
```

Alguns programas ainda não reconhecem o **flex** tentam executar seu predecessor, **lex**. Para dar suporte àqueles programas, crie um script com nome `lex` que chama o `flex` em modo de emulação do **lex**:

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex

exec /usr/bin/flex -l "$@"

# End /usr/bin/lex
EOF
chmod 755 /usr/bin/lex
```

6.29.2. Conteúdo do Flex

Programas instalados: flex, flex++ (link to flex) e lex
Biblioteca instalada: libfl.a

Descrição rápida

flex Uma ferramenta para gerar programas que reconhecem padrões em textos; ele permite a versatilidade de se especificar regras para localizar padrões, erradicando a necessidade de desenvolver um programa específico para isso

flex++ Invoca uma versão do **flex** que é usada exclusivamente por analisadores de código C++

lex Um script bash chamado lex que executa o **flex** em modo de emulação do **lex**

`libfl.a` A biblioteca `flex`

6.30. Gettext-0.14.3

O pacote Gettext contém utilitários para a internacionalização e localização. Eles permitem que os programas sejam compilados com suporte à língua nativa (NLS, Native Language Support) habilitando a exibição de mensagens de saída na língua nativa do usuário.

Tempo de compilação aproximado: 1.2 SBU
Espaço em disco necessário: 65.1 MB

Requisitos de instalação: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make e Sed

6.30.1. Instalação do Gettext

Prepare Gettext para a compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**. This takes a very long time, around 7 SBUs.

Instale o pacote:

```
make install
```

6.30.2. Conteúdo do Gettext

Programas instalados: autopoint, config.charset, config.rpath, envsubst, gettext, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext e xgettext
Bibliotecas instaladas: libasprintf.[a,so], libgettextlib.so, libgettextpo.[a,so] e libgettextsrc.so

Short Descriptions

autopoint	Copia arquivos infrastructure Gettext padrão em um pacote de fonte
config.charset	Exibe uma tabela de apelidos para codificação de caracteres
config.rpath	Exibe um conjunto de variáveis dependentes do sistema, descrevendo como configurar os caminhos de procura de bibliotecas compartilhadas em um executável
envsubst	Substitui variáveis de ambiente em strings no formato do shell
gettext	Traduz uma mensagem em linguagem natural para a língua do usuário buscando a tradução em um catálogo de mensagens
gettextize	Copia todos os arquivos gettext padrão para o diretório de um pacote, para iniciar sua internacionalização
hostname	Exibe o nome do computador da rede em vários formatos

msgattrib — Filtra as mensagens de um catálogo de tradução de acordo com os seus atributos

msgcat — Concatena e funde dois arquivos `.po`

msgcmp — Compara dois arquivos `.po` para verificar se ambos contêm o mesmo conjunto de strings do msgid

msgcomm — Encontra as mensagens que são comuns aos arquivos `.po`

msgconv — Converte um catálogo de tradução para uma codificação de caractere diferente

msgen — Cría um catálogo de tradução para o inglês

msgexec — Aplica um comando para todas as traduções de um catálogo

msgfilter — Aplica um filtro para todas as traduções de um catálogo

msgfmt — Compila uma tradução crua em código de máquina. É usado para criar o arquivo final de tradução do programa/pacote

msggrep — Extrai todas as mensagens de um catálogo de tradução que satisfazem um padrão dado ou pertencem a algum arquivo-fonte fornecido

msginit — Cria um novo arquivo `.po`, inicializando a meta-informação com valores do ambiente do usuário

msgmerge — Combina duas traduções cruas em um único arquivo

msgunfmt — Descompila arquivos de tradução em texto traduzido crú

msguniq — Unifica traduções duplicadas em um catálogo

ngettext — Exibe traduções em língua nativa de uma mensagem textual onde a forma gramatical depende de um número

xgettext — Extrai as linhas de mensagem de arquivos fonte para fazer o primeiro modelo de tradução

`libasprintf` — Define a classe *autosprintf*, que faz as rotinas de saída (output) formatadas em C utilizáveis em programas C++, para o uso com cadeias de caracteres *<string>* e fluxos *<iostream>*

`libgettextlib` — Uma biblioteca privada que contém as rotinas comuns usadas pelos vários programas do Gettext; ela não é para o uso geral

`libgettextpo` — Utilizada para escrever os programas especializados em processar arquivos `.po`; esta biblioteca é usada quando as aplicações que acompanham o Gettext (tais como **msgcomm**, **msgcmp**, **msgattrib** e **msgen**) não forem suficientes

`libgettextsrc` — Uma biblioteca privada que contém as rotinas comuns usadas pelos vários programas do Gettext; ela não é para o uso geral

6.31. Inetutils-1.4.2

O pacote Inetutils contém programas para o funcionamento básico em rede.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 8.7 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses e Sed

6.31.1. Instalação do Inetutils

O Inetutils tem problemas com a série 2.6 do kernel Linux. Repare isto aplicando o seguinte patch:

```
patch -Np1 -i ../inetutils-1.4.2-kernel_headers-1.patch
```

Nem todos os programas que vêm com o Inetutils serão instalados. Entretanto, o sistema de configuração do Inetutils insistirá em instalar todas as páginas man mesmo assim. O seguinte patch evitará isto:

```
patch -Np1 -i ../inetutils-1.4.2-no_server_man_pages-1.patch
```

Prepare o pacote para a compilação:

```
./configure --prefix=/usr --libexecdir=/usr/sbin \
    --sysconfdir=/etc --localstatedir=/var \
    --disable-logger --disable-syslogd \
    --disable-whois --disable-servers
```

Descrição das opções de configuração:

--disable-logger
Esta opção impede que Inetutils instale o programa **logger**, que é usado por scripts para passar mensagens para o System Log Daemon. Não instale este programa porque o Util-linux instalará uma versão melhor mais tarde.

--disable-syslogd
Esta opção impede que Inetutils instale o System Log Daemon, que será instalado com o pacote de Sysklogd.

--disable-whois
Esta opção incapacita a configuração do cliente **whois**, que está desatualizado. Instruções para um cliente **whois** melhor estão no livro de BLFS.

--disable-servers
Esta opção impede a instalação dos vários servidores de rede incluídos como a parte do pacote Inetutils. Estes servidores são considerados inadequados em um sistema básico LFS. Alguns são inseguros por natureza, ou seguros apenas em redes confiáveis. Mais informação podem ser encontradas em *http://www.linuxfromscratch.org/blfs/view/svn/basicnet/inetutils.html*. Note que estão disponíveis implementações melhores para muitos destes servidores.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Mova o programa **ping** para seu lugar compatível com o FHS:

```
mv /usr/bin/ping /bin
```

6.31.2. Conteúdo do Inetutils

Programas instalados: ftp, ping, rcp, rlogin, rsh, talk, telnet e tftp

Descrição rápida

ftp É o programa cliente do protocolo de transferência de arquivos

ping Envia pacotes-eco e apresenta relatórios sobre o tempo e o percurso das respostas na rede

rcp Executa a cópia de arquivo remoto

rlogin Executa um login remoto

rsh Executa um shell remoto

talk É usado para chat com um outro usuário

telnet Uma interface com o protocolo telnet

tftp Um programa básico para transferência de arquivos

6.32. IPRoute2-2.6.11-050330

O pacote IPRoute2 contém programas para funcionamento básico e avançado de redes baseadas no IPV4.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 4.3 MB

Requisitos de instalação: GCC, Glibc, Make, Linux-Headers e Sed

6.32.1. Instalação do IPRoute2

O binário **arpd** incluído neste pacote é requer o Berkeley DB. Como o **arpd** é uma exigência comum em um sistema Linux básico, remova a dependência do Berkeley DB aplicando o patch usando o comando abaixo. Se o binário **arpd** for necessário, as instruções para compilar o Berkeley DB podem ser encontradas no livro BLFS em *http://www.linuxfromscratch.org/blfs/view/svn/server/databases.html#db*.

```
patch -Np1 -i ../iproute2-2.6.11_050330-remove_db-1.patch
```

Prepare o IPRoute2 para a compilação:

```
./configure
```

Compile o pacote:

```
make SBINDIR=/sbin
```

Descrição das opções de configuração:

SBINDIR=/sbin
Assegura que os binários do IPRoute2 sejam instalados no diretório `/sbin`. Esta é a posição correta de acordo com o FHS, porque alguns dos binários do IPRoute2 são usados pelo pacote LFS-Bootscripts.

Instale o pacote:

```
make SBINDIR=/sbin install
```

6.32.2. Conteúdo do IPRoute2

Programas instalados: ctstat (link to lnstat), ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (link to lnstat), ss e tc.

Descrição rápida

ctstat	Utilitário de status da conexão
ifcfg	Um shell script para o comando **ip**
ifstat	Mostra as estatísticas da interface, incluindo a quantidade de pacotes transmitidos e recebidos pela interface

ip O principal executável. Tem diversas funções diferentes:

ip link ***`[dispositivo]`*** permite que os usuários vejam o estado dos dispositivos e façam mudanças

ip addr permite que os usuários vejam endereços e suas propriedades, adicionem endereços novos, e suprimam o velhos

ip neighbor permite que os usuários olhem emperramentos vizinhos e suas propriedades, adicionem entradas vizinhas novas, e suprimam o velhos

ip rule permite que os usuários vejam as políticas da distribuição e as mudem

ip route permite que os usuários vejam a tabela de distribuição e para modifiquem suas regras

ip tunnel permite que os usuários vejam e modifiquem os túneis do IP e suas propriedades

ip maddr permite que os usuários vejam e modifiquem os endereços multicast e suas propriedades, e

ip mroute permite que os usuários ajustem, mudem, ou suprimam a distribuição do multicast

ip monitor permite usuários monitorem continuamente o estado dos dispositivos, dos endereços e das rotas

lnstat Fornece as estatísticas da rede Linux. É uma versão mais completa para o antigo programa **rtstat**

nstat Mostra estatísticas da rede

routef Um componente do **ip route**. Esvazia as tabelas de distribuição

routel Um componente do **ip route**. Lista as tabelas de distribuição

rtacct Mostra o conteúdo do arquivo `/proc/net/rt_acct`

rtmon Utilitário de monitoramento de rota

rtpr Converte a saída do **ip -o** para um formato legível

rtstat Utilitário de status da rota

ss Similar ao comando **netstat**; mostra conexões ativas

tc Controlador de Tráfego Executável (Traffic Controlling Executable); implementação para Quality Of Service (QOS) e Class Of Service (COS)

tc qdisc permite que os usuários definam a disciplina da fila

tc class permite que os usuários definam as classes baseadas em programar da disciplina enfileirar-se

tc estimator permite que os usuários estimem o fluxo da rede em uma rede

tc filter permite que os usuários definam o filtro de pacotes do QOS/COS

tc policy permite que os usuários definam as políticas de QOS/COS

6.33. Perl-5.8.6

O pacote do Perl contém a "Practical Extraction and Report Language".

Tempo de compilação aproximado: 2.9 SBU
Espaço em disco necessário: 137 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make e Sed

6.33.1. Instalação do Perl

Para ter o controle total sobre a maneira como o Perl é configurado, execute o script interativo **Configure** e manualmente diga a maneira como este pacote é configurado. Se os padrões de auto-detecção forem apropriados, prepare o Perl para a compilação com:

```
./configure.gnu --prefix=/usr -Dpager="/bin/less -isR"
```

Descrição das opções de configuração:

`-Dpager="/bin/less -isR"`
Isto corrige um erro no código **perldoc** invocando o programa **less**.

Compile o pacote:

```
make
```

Para executar o conjunto de testes, crie primeiro um arquivo básico `/etc/hosts` é necessário para alguns dos testes:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

Execute agora os testes, se quiser:

```
make test
```

Instale o pacote:

```
make install
```

6.33.2. Conteúdo do Perl

Programas instalados: a2p, c2ph, dprofpp, enc2xs, find2perl, h2ph, h2xs, libnetcfg, perl, perl5.8.6 (vínculo to perl), perlbug, perlcc, perldoc, perlivp, piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, psed (vínculo to s2p), pstruct (vínculo to c2ph), s2p, splain e xsubpp
Bibliotecas instaladas: São centenas e não serão listados aqui

Descrição rápida

a2p	Conversor de AWK para PERL
c2ph	Exibe as estruturas C como se fossem geradas a partir de **cc -g -S**

dprofpp Exibe dados de perfil do Perl

en2cxs Configura uma extensão do Perl para o módulo Encode do Unicode Character Mappings ou Tcl Encoding Files

find2perl Converte linhas de comando do **find** para código Perl

h2ph Converte arquivos de cabeçalhos `.h` do C para arquivos de cabeçalhos `.ph` do Perl.

h2xs Converte arquivos de cabeçalhos `.h` para extensões do Perl

libnetcfg Pode ser usado para configurar o `libnet`

perl Ela combina algumas das melhores funcionalidades do C, **sed**, **awk** e **sh** em uma única linguagem poderosa

perl5.8.6 Um hardlink ao **perl**

perlbug Auxilia a geração de relatórios de bug sobre o Perl ou seus módulos e os envia por e-mail

perlcc Gera executáveis a partir de programas Perl

perldoc Apresenta uma parte da documentação em formato .pod encontrada nos diretórios do Perl ou em um script Perl e a exibe

perlivp A Perl Installation Verification Procedure; pode ser usada para verificar se o Perl e suas bibliotecas foram instalados corretamente

piconv Uma versão do Perl do **iconv**

pl2pm Uma ferramenta que auxilia na conversão de bibliotecas `.pl` estilo Perl4 para módulos `.pm` do Perl5

pod2html Converte arquivos do formato pod para o HTML

pod2latex Converte arquivos do formato pod para o LaTeX

pod2man Converte arquivos do formato pod para o *roff

pod2text Converte arquivos do formato pod para texto ASCII formatado

pod2usage Exibe mensagens de uso para documentos pod embutidos em arquivos

podchecker Verifica a sintaxe de arquivos de documentação em formato pod

podselect Exibe seções selecionadas da documentação pod na saída padrão

psed Uma versão do Perl do editor de stream **sed**

pstruct Exibe estruturas C como se fossem geradas a partir **cc -g -S**

s2p Conversor de scripts **sed** para Perl

splain Programa que força diagnósticos detalhados de avisos no Perl

xsubpp Converte código Perl XS em código C

6.34. Texinfo-4.8

O pacote de Texinfo contém programas para a leitura, a escrita, e conversão de páginas info.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 14.7 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses e Sed

6.34.1. Instalação do Texinfo

Prepare o Texinfo para a compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

Opcionalmente, instale os componentes que pertencem a uma instalação do TeX:

```
make TEXMF=/usr/share/texmf install-tex
```

Descrição dos parâmetros do make:

`TEXMF=/usr/share/texmf`
A variável do makefile `TEXMF` armazena a posição da raiz da árvore do TeX se, por exemplo, o pacote TeX for instalado mais tarde.

O sistema de documentação Info utiliza um arquivo texto puro para armazenar sua lista de entradas de menu. O arquivo é encontrado no diretório `/usr/share/info/dir`. Infelizmente, devido a problemas ocasionais nos makefiles de vários pacotes, ele pode às vezes sair da sincronização com as páginas do info instaladas no sistema. Se o arquivo `/usr/share/info/dir` precisar ser recriado, os seguintes comandos opcionais realizam esta tarefa:

```
cd /usr/share/info
rm dir
for f in *
do install-info $f dir 2>/dev/null
done
```

6.34.2. Conteúdo do Texinfo

Programas instalados: info, infokey, install-info, makeinfo, texi2dvi e texindex

Descrição rápida

info	Usado para ler páginas info que são parecidas com páginas man mas geralmente vão muito mais a fundo do que simples explicações das opções de linha de comando. Por exemplo, compare **man bison** e **info bison**.
infokey	Compiles a source file containing Info customizations into a binary format
install-info	Usado para instalar páginas info; ele atualiza as entradas nos arquivos de índices **info**
makeinfo	Converte documentos-fontes Texinfo para os formatos: arquivos info, texto plano e HTML
texi2dvi	Formata um documento Texinfo em um documento independente do dispositivo que pode ser impresso
texindex	Usado para ordenar arquivos-índices Texinfo

6.35. Autoconf-2.59

O pacote de Autoconf contém programas para gerar shell scripts que podem automaticamente configurar códigos fonte.

Tempo de compilação aproximado: 0.5 SBU
Espaço em disco necessário: 8.5 MB

Requisitos de instalação: Bash, Coreutils, Diffutils, Grep, M4, Make, Perl e Sed

6.35.1. Instalação do Autoconf

Prepare o Autoconf para a compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**. Isto faz exame longo, aproximadamente 2 SBUs.

Instale o pacote:

```
make install
```

6.35.2. Conteúdo do Autoconf

Programas instalados: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate e ifnames

Descrição

autoconf Produz os shell scripts que configuram automaticamente pacotes de códigos fonte de software para se adaptar a muitos tipos de sistemas Unix. Os scripts de configuração que produz são independentes—sua execução não requer o programa **autoconf**.

autoheader Cria um arquivo-modelo de declarações *#define* em linguagem C para uso com o configure

autom4te Usado para chamar o o processador de macro M4

autoreconf Automaticamente executa o **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize** e o **libtoolize** na ordem correta para economizar tempo quando são feitas mudanças nos arquivos modelo **autoconf** e **automake**

autoscan Auxilia na criação de um arquivo `configure.in` para um pacote de software; examina os arquivos fonte em uma árvore do diretório, procura os recursos comuns de portabilidade, e cria um arquivo `configure.scan` que sirva como um arquivo `configure.in` preliminar para o pacote

autoupdate Modifica um arquivo `configure.in` que ainda chama macros **autoconf** por seus nomes antigos para que use os nomes de macro atuais

ifnames Ajuda a escrever o arquivo `configure.in` para um pacote de software; mostra os identificadores que o pacote usa em condicionai de preprocessamento C. Se um pacote já configurado para ter algum portabilidade, este programa pode ajudar a determinar o que no **configure** precisam ser verificado. Pode também preencher lacunas em um arquivo `configure.in` gerado pelo **autoscan**

6.36. Automake-1.9.5

O pacote Automake contém programas para gerar makefiles para uso com o Autoconf.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 8.8 MB

Requisitos de instalação: Autoconf, Bash, Coreutils, Diffutils, Grep, M4, Make, Perl e Sed

6.36.1. Instalação do Automake

Prepare o Automake para a compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**. Isto faz um exame longo, aproximadamente 5 SBUs.

Instale o pacote:

```
make install
```

6.36.2. Conteúdo do Automake

Programas instalados: acinstall, aclocal, aclocal-1.9.5, automake, automake-1.9.5, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree e ylwrap

Descrição rápida

acinstall	Um script que instala arquivos M4 tipo aclocal (aclocal-style).
aclocal	Gera os arquivos `aclocal.m4` baseado no conteúdo de arquivos `configure.in`
aclocal-1.9.5	Um hard link para o **aclocal**
automake	Uma ferramenta para gerar automaticamente arquivos `Makefile.in` a partir de arquivos `Makefile.am`. Para criar todos os arquivos `Makefile.in` para um pacote, execute este programa no diretório de nível mais alto, sem argumentos. O automake vai analisar o arquivo `configure.in`, localizar automaticamente cada arquivo `Makefile.am` e gerar o arquivo `Makefile.in` correspondente
automake-1.9.5	Um hard link para o **automake**
compile	Um script que age como um intermediário para compiladores
config.guess	Um script que detecta o tipo de sistema
config.sub	Um script de validação da configuração

depcomp Um script que gera dependências a partir dos efeitos da compilação de programas-teste

elisp-comp Um script que compila arquivos .el (Emacs Lisp code).

install-sh Um script que instala um programa, outro script ou um arquivo de dados

mdate-sh Um script que imprime a data de modificação de um arquivo ou diretório

missing Um script que age como um mediador para alguns programas que estão faltando no sistema

mkinstalldirs Um script que cria uma hierarquia de diretórios

py-compile Compila um programa de Python

symlink-tree Um script para criar uma árvore de symlink de uma árvore de diretório

ylwrap Um script que funciona como mediador para requisições **lex** e **yacc**

6.37. Bash-3.0

O pacote bash contém o shell Bourne-Again SHell.

Tempo de compilação aproximado: 1.2 SBU
Espaço em disco necessário: 20.6 MB

Requisitos de instalação: Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses e Sed.

6.37.1. Instalação do Bash

O seguinte patch repara várias coisas, incluindo um problema onde o bash às vezes mostra somente 33 caracteres em uma linha, quebrando então para a seguinte:

```
patch -Np1 -i ../bash-3.0-fixes-3.patch
```

O bash apresenta também problemas quando compilado com algumas versões mais novas do Glibc. O seguinte patch resolve isto:

```
patch -Np1 -i ../bash-3.0-avoid_WCONTINUED-1.patch
```

Prepare o bash para a compilação:

```
./configure --prefix=/usr --bindir=/bin \
    --without-bash-malloc --with-installed-readline
```

Descrição das opções de configuração:

--with-installed-readline
Este as opções dizem o bash para usar a biblioteca do `readline` é instalada já no sistema melhor que em usar sua própria versão do readline.

Compile o pacote:

```
make
```

Para testar os resultados, use: **make tests**.

Instale o pacote:

```
make install
```

Execute o programa **bash** recém-compilado (que substitui esse que está sendo executado atualmente):

```
exec /bin/bash --login +h
```

Note

Os parâmetros usados fazem o **bash** processar um shell interativo (login-shell) mantendo desabilitado o hashing de modo que os programas novos sejam encontrados assim que se tornem disponíveis.

6.37.2. Conteúdo do Bash

Programas instalados: bash, bashbug e sh (vínculo para o bash)

Descrição rápida

bash bash é o Bourne-Again SHell, um interpretador de comandos largamente utilizado em sistemas Unix. O programa bash lê da entrada padrão (o teclado). Um usuário digita algo, o programa avalia o que ele digitou e realiza alguma ação, como executar um programa

bashbug Um script shell que ajuda o usuário a compor e enviar relatórios de bugs relacionados ao **bash**, por meio de um método padronizado

sh Um vínculo simbólico para o programa **bash**; Quando invocado como **sh**, **bash** o bash tenta emular o funcionamento de versões históricas do **sh** o mais idêntico possível, ao mesmo tempo que se mantém adequado ao padrão POSIX

6.38. File-4.13

O pacote File contém um utilitário para determinar o tipo de um ou vários arquivos.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 6.2 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Sed e Zlib

6.38.1. Instalação do File

Prepare o File para a compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

6.38.2. Conteúdo do File

Programas instalados: file
Biblioteca instalada: libmagic.[a,so]

Descrição rápida

file	Tenta classificar cada arquivo especificado; faz isso efetuando diversos testes—testes de sistemas de arquivos, testes de números mágicos, testes de linguagem
`libmagic`	Contém as rotinas para o reconhecimento do número má, usadas pelo programa **file**

6.39. Libtool-1.5.14

O pacote Libtool contém o script genérico GNU de suporte à biblioteca. Facilita o uso de bibliotecas compartilhadas através de uma interface consistente e portável.

Tempo de compilação aproximado: 1.5 SBU
Espaço em disco necessário: 19.7 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make e Sed

6.39.1. Instalação do Libtool

Prepare o Libtool para a compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

6.39.2. Conteúdo do Libtool

Programas instalados: libtool e libtoolize
Bibliotecas instaladas: libltdl.[a,so]

Descrição rápida

libtool	Fornece serviços genéricos de suporte à compilação de bibliotecas
libtoolize	Fornece um meio padronizado para adicionar suporte ao **libtool** em um pacote
`libltdl`	Uma pequena biblioteca que esconde dos programadores as várias dificuldades no desenvolvimento de bibliotecas

6.40. Bzip2-1.0.3

O pacote Bzip2 contém programas para compressão e descompressão de arquivos. Arquivos de texto comprimindo com o **bzip2** alcançam uma porcentagem muito melhor de compressão do que com o tradicional **gzip**.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 3.9 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc e Make

6.40.1. Instalação do Bzip2

Prepare o Bzip2 para a compilação com:

```
make -f Makefile-libbz2_so
make clean
```

O parâmetro *-f* faz com que o Bzip2 seja configurado usando um arquivo diferente do `Makefile`, neste caso o arquivo `Makefile-libbz2_so`, que cria a biblioteca dinâmica `libbz2.so` e cria as ligações simbólicas dos utilitários Bzip2 com ela.

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make test`**.

Se estiver reinstalando o Bzip2, execute antes o comando **`rm -f /usr/bin/bz*`** os passos seguintes **make install** irão falhar.

Instale os programas:

```
make install
```

Instale o binário compartilhado **bzip2** no `/bin` e faça algumas ligações simbólicas necessárias, e apague alguns arquivos:

```
cp bzip2-shared /bin/bzip2
cp -a libbz2.so* /lib
ln -s ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm /usr/bin/{bunzip2,bzcat,bzip2}
ln -s bzip2 /bin/bunzip2
ln -s bzip2 /bin/bzcat
```

6.40.2. Conteúdo do Bzip2

Programas instalados: bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless e bzmore
Bibliotecas instaladas: libbz2.[a,so]

Descrição rápida

bunzip2 Descompacta arquivos bzip2

bzcat Descompacta todos os arquivos fornecidos para a saída padrão

bzcmp Executa o programa **cmp** em arquivos compactados

bzdiff Executa o programa **diff** em arquivos compactados

bzgrep Executa o programa **grep** em arquivos compactados

bzegrep Executa o programa **egrep** em arquivos compactados

bzfgrep Executa o programa **fgrep** em arquivos compactados

bzip2 Compacta arquivos usando o algoritmo Burrows-Wheeler de compressão de texto através da ordenação de blocos. A compactação é consideravelmente melhor que a dos compactadores convencionais baseados no algoritmo “Lempel-Ziv”, como o **gzip**

bzip2recover Tenta recuperar dados de arquivos compactados bzip2 danificados

bzless Executa o programa **less** em arquivos compactados

bzmore Executa o programa **more** em arquivos compactados

`libbz2*` Biblioteca para implementação de compressão de dados usando o algoritmo Burrows-Wheeler

6.41. Diffutils-2.8.1

O pacote Diffutils contém programas que mostram as diferenças entre arquivos ou diretórios.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 5.6 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Sed

6.41.1. Instalação do Diffutils

Prepare o Diffutils para a compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Este pacote não vem com um conjunto de testes.

Instale o pacote:

```
make install
```

6.41.2. Conteúdo do Diffutils

Programas instalados: cmp, diff, diff3 e sdiff

Descrição rápida

cmp Compara dois arquivos e relata se ou em que bytes diferem

diff Compara dois arquivos ou diretórios e relata em que linhas os arquivos diferem

diff3 Compara três arquivos linha por linha

sdiff Mescla dois arquivos e exibe os resultados interativamente

6.42. Kbd-1.12

O pacote Kbd contém arquivos de mapeamento de teclas e utilitários para o teclado.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 11.8 MB

Requisitos de instalação: Bash, Binutils, Bison, Coreutils, Diffutils, Flex, GCC, Gettext, Glibc, Grep, Gzip, M4, Make e Sed

6.42.1. Instalação do Kbd

Prepare o Kbd para a compilação:

```
./configure
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

6.42.2. Conteúdo do Kbd

Programas instalados: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, getunimap, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (vínculo to psfxtable), psfgettable (vínculo to psfxtable), psfstriptable (vínculo to psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setlogcons, setmetamode, setvesablank, showconsolefont, showkey, unicode_start e unicode_stop

Descrição rápida

chvt	Modifica o terminal virtual de primeiro plano
deallocvt	Libera terminais virtuais não usados
dumpkeys	Exibe as tabelas de conversão do teclado
fgconsole	Exibe o número do terminal virtual ativo
getkeycodes	Exibe a tabela de mapeamento scancode-para-keycode do kernel
getunimap	Exibe o unimap que está sendo usado
kbd_mode	Exibe ou configura o modo do teclado
kbdrate	Ajusta a velocidade de repetição e espera do teclado
loadkeys	Carrega as tabelas de conversão do teclado
loadunimap	Carrega a tabela de mapeamento unicode-para-fonte do kernel

mapscrn	Um programa obsoleto que carrega a tabela de saída de caractere definida pelo usuário dentro do driver do console. Isto é feito atualmente pelo **setfont**
openvt	Inicia um programa em um novo terminal virtual (VT).
psfaddtable	Um vínvulo para o **psfxtable**
psfgettable	Um vínvulo para o **psfxtable**
psfstriptable	Um vínvulo para o **psfxtable**
psfxtable	Manipula as tabelas de caracteres Unicode para fontes do console
resizecons	Modifica a configuração do kernel para o tamanho do console
setfont	Modificar as fontes EGA/VGA no console
setkeycodes	Carrega as entradas da tabela de mapeamento scancode-para-keycode do kernel
setleds	Configura os LEDs do teclado. Muitas pessoas acham isto útil para ter o numlock habilitado por padrão
setlogcons	Envia mensagens do kernel para o console
setmetamode	Define o funcionamento da tecla alt (meta) do teclado
setvesablank	Permite a você configurar o protetor de tela do hardware (uma tela em branco)
showconsolefont	Exibe a atual fonte EGA/VGA de tela do console
showkey	Mostra os scancodes, os keycodes, e os códigos ASCII das teclas pressionadas no teclado
unicode_start	Põe o teclado e o console no modo UNICODE. Não use em um sistema LFS, pois as aplicações não estão configuradas para suportar o UNICODE
unicode_stop	Retira o teclado e o console do modo Unicode

6.43. E2fsprogs-1.37

O pacote de E2fsprogs contém utilitários para manipular o sistema de arquivos `ext2`. Suporta também o sistema de arquivos `ext3` journaling.

Tempo de compilação aproximado: 0.6 SBU
Espaço em disco necessário: 40.0 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed e Texinfo

6.43.1. Instalação do E2fsprogs

Repare um erro de compilação no conjunto de testes do E2fsprogs:

```
sed -i -e 's/-DTEST/$(ALL_CFLAGS) &/' lib/e2p/Makefile.in
```

Recomenda-se que o E2fsprogs seja configurado em um subdiretório da árvore dos arquivos fonte:

```
mkdir build
cd build
```

Prepare o E2fsprogs para a compilação:

```
../configure --prefix=/usr --with-root-prefix="" \
    --enable-elf-shlibs --disable-evms
```

Descrição das opções de configuração:

--with-root-prefix=""
Certos programas e bibliotecas (tais como o programa **e2fsck**) são considerados essenciais ao sistema. Quando, por exemplo, `/usr` não está montado, eles precisam estar disponíveis. Estes arquivos são instalados em diretórios como `/lib` e `/sbin`. Se esta opção não for passada ao script configure do E2fsprogs, eles serão instalados em `/usr`, algo que não desejamos.

--enable-elf-shlibs
Cria as bibliotecas compartilhadas usadas por alguns programas deste pacote.

--disable-evms
Isto desabilita a configuração do plugin Enterprise Volume Management System (EVMS). Este plugin não está atualizado com as interfaces internas mais recentes do EVMS e o EVMS não é instalado como a parte de um sistema básico LFS, assim este plugin não é necessário. Veja o website de EVMS em *http://evms.sourceforge.net/* para mais informação a respeito do EVMS.

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale os binários e a documentação:

```
make install
```

Instale as bibliotecas compartilhadas:

```
make install-libs
```

6.43.2. Conteúdo do E2fsprogs

Programas instalados: badblocks, blkid, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, findfs, fsck, fsck.ext2, fsck.ext3, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs e uuidgen.
Bibliotecas instaladas: libblkid.[a,so], libcom_err.[a,so], libe2p.[a,so], libext2fs.[a,so], libss.[a,so] e libuuid.[a,so]

Descrição rápida

badblocks	Procura blocos danificados em um dispositivo (normalmente uma partição do disco)
blkid	Um utilitário de linha de comando para encontrar e exibir atributos do dispositivo de blocos
chattr	Modifica os atributos de arquivos em sistemas de arquivos `ext2` e `ext3` (versão com journaling do `ext2`)
compile_et	Um compilador da tabela de erro; lista nomes de códigos de erro e as respectivas mensagens em um arquivo-fonte C próprio para uso com a biblioteca `com_err`
debugfs	Um depurador do sistema de arquivos. Ele pode ser usado para examinar e modificar o estado de um sistema de arquivos `ext2`
dumpe2fs	Exibe informações de grupo sobre o superbloco e demais blocos para o sistema de arquivos presente em um dispositivo específico
e2fsck	É usado verificar, e opcionalmente reparar sistemas de arquivos `ext2` e `ext3`
e2image	É usado conservar dados críticos do sistema de arquivos `ext2` em um arquivo
e2label	Exibe ou modifica a etiqueta do sistema de arquivos `ext2` localizado no dispositivo especificado
findfs	Encontra um sistema de arquivos pela etiqueta ou pelo identificador universal original (UUID)
fsck	É usado verificar, e opcionalmente reparar sistemas de arquivos
fsck.ext2	Por padrão, verifica sistemas de arquivos `ext2`
fsck.ext3	Por padrão, verifica sistemas de arquivos `ext3`
logsave	Registra a saída de um comando em um arquivo de log
lsattr	Lista os atributos de arquivos em um sistema de arquivos estendido
mk_cmds	Recebe um arquivo contendo uma tabela de comandos como entrada e produz um

arquivo-fonte C como saída, próprio para ser usado com a biblioteca de subsistema `libss`

mke2fs Cria um sistema de arquivos ext2 ou ext3 no dispositivo dado

mkfs.ext2 Cria sistemas de arquivos `ext2`

mkfs.ext3 Cria sistemas de arquivos `ext3`

mklost+found Usado para criar um diretório `lost+found` em um sistema de arquivos Linux `ext2`; reserva previamente blocos de disco para o diretório, para torná-lo útil ao **e2fsck**

resize2fs Pode ser usado ampliar ou reduzir um sistema de arquivos `ext2`

tune2fs Configura parâmetros ajustáveis em sistemas de arquivos Linux `ext2`

uuidgen Cria um novo identificador universalmente único (universally unique identifier, UUID), usando a biblioteca libuuid. O novo UUID pode razoavelmente ser considerado único entre todos os UUIDs criados, no sistema local e em outros sistemas, no passado e no futuro

`libblkid` Contem rotinas para a identificação de dispositivo e a extração do símbolo

`libcom_err` A rotina comum de exposição de erro

`libe2p` Usada por **dumpe2fs**, **chattr** e **lsattr**

`libext2fs` Contem rotinas para permitir que programas em nível de usuário manipulem um sistema de arquivos `ext2`

`libss` Usado por **debugfs**

`libuuid` Contem rotinas para gerar identificadores originais para os objetos que podem ser acessíveis além do sistema local

6.44. Grep-2.5.1a

O pacote Grep contém programas para procurar em arquivos. É usado para exibir linhas de um arquivo que satisfazem determinado padrão.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 4.5 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Make, Sed e Texinfo

6.44.1. Instalação de Grep

Prepare o Grep para a compilação:

```
./configure --prefix=/usr --bindir=/bin
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

6.44.2. Conteúdo do Grep

Programas instalados: egrep (link para grep), fgrep (vínculo para grep) e grep

Descrição rápida

egrep Exibe linhas de arquivos que satisfazem a um padrão estendido de expressão regular

fgrep Exibe linhas de arquivos que satisfazem a uma lista fixa de strings, separadas por novas linhas (newlines)

grep Exibe linhas de arquivos que satisfazem a um padrão básico de expressão regular

6.45. GRUB-0.96

O pacote GRUB contém o GRand Unified Bootloader.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 10.0 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses e Sed

6.45.1. Instalação do GRUB

Este pacote apresenta conhecidos problemas quando suas opções de otimização padrão (incluindo as opções *-march* e *-mcpu*)são modificadas. Se alguma variável de ambiente modificar estas opções de otimização, como por exemplo `CFLAGS` e `CXXFLAGS`, remova-as enquanto configura o GRUB.

Prepare o GRUB para a compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **make check**.

Note que os resultados deste teste mostrarão sempre a mensagem de erro "ufs2_stage1_5 is too big". Isto é devido a uma certa característica do compilador, mas pode ser ignorado a menos que você planeje inicializar de uma partição UFS. Estas partições normalmente são usadas somente por estações de trabalho da Sun.

Instale o pacote:

```
make install
mkdir /boot/grub
cp /usr/lib/grub/i386-pc/stage{1,2} /boot/grub
```

Substitua `i386-pc` pelo diretório apropriado para o hardware em uso.

O diretório `i386-pc` contém um certo número de arquivos `*stage1_5`, que são diferentes para sistemas de arquivos diferentes. Veja os arquivos disponíveis e copie os apropriados para o diretório `/boot/grub`.A maioria de usuários copiará os arquivos `e2fs_stage1_5` e/ou `reiserfs_stage1_5`.

6.45.2. Conteúdo do GRUB

Programas instalados: grub, grub-install, grub-md5-crypt, grub-terminfo e mbchk

Descrição rápida

grub	O shell de comandos do The Grand Unified Bootloader
grub-install	Instala o GRUB no dispositivo informado
grub-md5-crypt	Encripta uma senha no formato MD5

grub-terminfo Gera um comando terminfo a partir de um nome terminfo; pode ser empregado se um terminal desconhecido estiver sendo usado

mbchk Verifica o formato de um kernel multi-boot

6.46. Gzip-1.3.5

O pacote Gzip contém programas para comprimir e descomprimir arquivos.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 2.2 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make e Sed

6.46.1. Instalação do Gzip

Gzip tem 2 vulnerabilidades de segurança conhecidas. O seguinte patch elimina ambas:

```
patch -Np1 -i ../gzip-1.3.5-security_fixes-1.patch
```

Prepare o Gzip para a compilação:

```
./configure --prefix=/usr
```

O script **gzexe** tem a posição do binário **gzip** incorporada nele. Como a posição do binário será alterada mais tarde, o seguinte comando assegura que a nova posição seja inscrita no script:

```
sed -i 's@"BINDIR"@/bin@g' gzexe.in
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Mova o programa **gzip** para o diretório `/bin` e crie algumas ligações simbólicas:

```
mv /usr/bin/gzip /bin
rm /usr/bin/{gunzip,zcat}
ln -s gzip /bin/gunzip
ln -s gzip /bin/zcat
ln -s gzip /bin/compress
ln -s gunzip /bin/uncompress
```

6.46.2. Conteúdo do Gzip

Programas instalados: compress (vínculo para gzip), gunzip (vínculo para gzip), gzexe, gzip, uncompress (vínculo to gunzip), zcat (vínculo to gzip), zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore e znew

Descrição rápida

compress	Compacta arquivos
gunzip	Descompacta arquivos que são compactados com o gzip

gzexe Compacta executáveis que são descompactados automaticamente ao serem executados (ao custo de perda de performance)

gzip Compacta arquivos usando a codificação Lempel-Ziv (LZ77)

uncompress Descompacta arquivos compactados

zcat Descompacta e escreve para a saída padrão uma lista de arquivos ou um arquivo sendo lido da entrada padrão

zcmp Executa o programa **cmp** em arquivos compactados com o gzip

zdiff Executa o programa **diff** em arquivos compactados com o gzip

zegrep Executa o programa **egrep** em arquivos compactados com o gzip

zfgrep Executa o programa **fgrep** em arquivos compactados com o gzip

zforce Força uma extensão `.gz` em todos os arquivos gzip, assim o **gzip** não irá compactá-los duas vezes. Isto pode ser útil para arquivos com nomes truncados após uma transferência de arquivos

zgrep Executa o programa **grep** em arquivos compactados com o gzip

zless Executa o programa **less** em arquivos compactados com o gzip

zmore Executa o programa **more** em arquivos compactados com o gzip

znew Recompacta arquivos no formato **compress** para o formato **gzip**— `.Z` para `.gz`

6.47. Hotplug-2004_09_23

O pacote Hotplug contém os scripts que reagem aos eventos hotplug gerados pelo kernel. Tais eventos correspondem a cada mudança no estado do kernel visível no sistema de arquivos `sysfs`, como por exemplo a adição e a remoção de algum hardware. Este pacote também detecta o hardware existente durante o boot e carrega os módulos pertinentes no kernel em execução.

Tempo de compilação aproximado: 0.01 SBU
Espaço em disco necessário: 460 KB

Requisitos de instalação: Bash, Coreutils, Find, Gawk e Make

6.47.1. Instalação do Hotplug

Instale o pacote Hotplug:

```
make install
```

Copie um arquivo que o “install” omite.

```
cp etc/hotplug/pnp.distmap /etc/hotplug
```

Remova o script de inicialização que Hotplug instala, pois nós vamos usar o script incluído no pacote LFS-Bootscripts:

```
rm -rf /etc/init.d
```

Dispositivos de rede com hotplug ainda não são suportados pelo pacote LFS-Bootscripts. Por esta razão, remova o agente hotplug de rede:

```
rm -f /etc/hotplug/net.agent
```

Crie um diretório para armazenar os firmwares que podem ser carregados pelo **hotplug**:

```
mkdir /lib/firmware
```

6.47.2. Conteúdo do Hotplug

Programa instalado: hotplug
Scripts instalados: /etc/hotplug/*.rc, /etc/hotplug/*.agent
Arquivos instalados: /etc/hotplug/hotplug.functions, /etc/hotplug/blacklist, /etc/hotplug/{pci,usb}, /etc/hotplug/usb.usermap, /etc/hotplug.d e /var/log/hotplug/events

Descrição rápida

hotplug	Este script é chamado por padrão pelo kernel Linux quando algo muda em seu estado interno (por exemplo, um dispositivo novo é adicionado ou um dispositivo existente é removido)
/etc/hotplug/*.rc	Estes scripts são usados para plugging frio, isto é, detectando e agindo em cima de hardware presente durante a inicialização do sistema. São chamados pelo script de inicialização `hotplug` incluído no pacote LFS-Bootscripts. Os scripts ***.rc** tentam recuperar os eventos hotplug que foram perdidos durante o boot do sistema porque, por o exemplo, o sistema de arquivos root não foi montado pelo kernel
/etc/hotplug/*.agent	Estes scripts são chamados pelo **hotplug** em resposta aos diferentes tipos de eventos hotplug gerados pelo kernel. Sua ação deve carregar os módulos correspondentes do kernel e chamar qualquer script fornecido pelo usuário
`/etc/hotplug/blacklist`	Este arquivo contém a lista dos módulos que nunca devem ser carregados no kernel pelos scripts Hotplug
`/etc/hotplug/hotplug.functions`	Este arquivo contém as funções comuns usadas por outros scripts do pacote Hotplug
`/etc/hotplug/{pci,usb}`	Estes diretórios contém os alimentadores escritos pelo usuário para eventos hotplug
`/etc/hotplug/usb.usermap`	Este arquivo contém as regras que determinam que alimentadores definidos pelo usuário chamar para cada dispositivo USB, baseado em seu vendor ID e outros atributos
`/etc/hotplug.d`	Este diretório contém os programas (ou as ligações simbólicas para eles) que estão interessados em receber eventos do hotplug. Por o exemplo, o Udev põe sua ligação simbólica aqui durante a instalação
`/lib/firmware`	Este diretório contém os firmwares para os dispositivos que necessitam ter seus firmwares carregados antes do uso
`/var/log/hotplug/events`	Este arquivo contém todos os eventos em que o **hotplug** foi chamado desde o boot

6.48. Man-1.5p

O pacote Man contém programas para encontrar e visualizar as páginas man.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 2.9 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make e Sed

6.48.1. Instalação do Man

Dois ajustes precisam ser feitos ao fontes do Man.

O primeiro é utilizar a substituição pelo **sed** para adicionar o parâmetro *-R* à variável `PAGER` modo que as seqüências de escape sejam manipuladas corretamente pelo Less:

```
sed -i 's@-is@&R@g' configure
```

O segundo também utiliza a substituição pelo **sed**, para comentar a linha “MANPATH /usr/man” no arquivo `man.conf` para impedir resultados redundantes ao usar alguns programas, como o **whatis**:

```
sed -i 's@MANPATH./usr/man@#&@g' src/man.conf.in
```

Prepare o Man para a compilação:

```
./configure -confdir=/etc
```

Descrição das opções de configuração:

-confdir=/etc
Diz ao programa **man** para procurar o arquivo de configuração `man.conf` no diretório `/etc`.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Note

Se você estiver trabalhando em um terminal que não suporte atributos de texto tais como cor e bold (realce), você pode desabilitar o Select Graphic Rendition (SGR) editando o arquivo `man.conf` para acrescentar a opção *-c* à variável `NROFF`. Se você usar múltiplos terminais para um computador, pode ser melhor adicionar seletivamente a variável de ambiente `GROFF_NO_SGR` nos terminais que não suportam SGR.

Se o conjunto de caracteres do locale usar caracteres de 8-bit, procure pela linha que começa com o "NROFF" em `/etc/man.conf`, e verifique se é igual a esta:

```
NROFF  /usr/bin/nroff -Tlatin1 -mandoc
```

Note que "latin1" é utilizado mesmo não sendo o conjunto de caracteres do locale. A razão disto é porque, conforme suas especificações, o **groff** o groff não lida com definições estranhas ao padrão International Organization for Standards (ISO) 8859-1 sem alguns códigos de escape incomuns. Ao formatar páginas do man, o **groff** assume que elas estão codificadas como ISO 8859-1 e esta opção *`-Tlatin1`* diz ao **groff** para usar este encoding para a saída. Como o **groff**, em razão disto, não faz nenhuma modificação no encoding de entrada, como resultado o texto formatado na saída estará com exatamente o mesmo encoding de entrada, e conseqüentemente será usável normalmente como a entrada para um pager.

Isto não resolve o problema para o não funcionamento do programa **man2dvi** com páginas localizadas do man nos locales não-ISO 8859-1. Também ele não trabalha com multibyte character sets. O primeiro problema não tem atualmente uma solução. E o segundo problema não é do nosso interesse porque a instalação do LFS não suporta multibyte character sets.

Informações adicionais sobre a compressão das páginas man e info podem ser encontradas no livro BLFS em *http://www.linuxfromscratch.org/blfs/view/cvs/postlfs/compressdoc.html*.

6.48.2. Conteúdo do Man

Programas instalados: apropos, makewhatis, man, man2dvi, man2html e whatis

Descrição rápida

apropos	Procura a base de dados **whatis** e apresenta as descrições rápidas dos comandos do sistema que contenha uma determinada sequência de caracteres
makewhatis	Compila a base de dados **whatis**; lê todas as páginas de manual contidas nas seções do `MANPATH` escreve uma linha na base de dados do **whatis**
man	Formata e exibe as páginas de manual on-line
man2dvi	Converte uma página de manual para o formato dvi
man2html	Converte uma página de manual para HTML
whatis	Faz uma busca na base de dados **whatis** e apresenta a descrição rápida dos comandos do sistemas que contenham a palavra fornecida inteira

6.49. Make-3.80

O pacote make contém um programa para compilar pacotes.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 7.1 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep e Sed

6.49.1. Instalação do Make

Prepare o make para a compilação:

```
./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **make check**.

Instale o pacote:

```
make install
```

6.49.2. Conteúdo do Make

Programa instalado: make

Descrição rápida

make Determina automaticamente que partes de um pacote necessitam ser (re)compiladas e emite então os comandos apropriados

6.50. Module-Init-Tools-3.1

O pacote Module-Init-Tools contém programas para manipular módulos nos kernel Linux de versão maior ou igual à 2.5.47.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 4.9 MB

Requisitos de instalação: Bash, Binutils, Bison, Coreutils, Diffutils, Flex, GCC, Glibc, Grep, M4, Make e Sed

6.50.1. Instalação do Module-Init-Tools

O Module-Init-Tools tenta reescrever sua página man `modprobe.conf` durante o processo de configuração. Isto não é necessário e requer também o **docbook2man** — que não é instalado pelo LFS. Executar o seguinte comando evitar isto:

```
touch modprobe.conf.5
```

Prepare o Module-Init-Tools para a compilação:

```
./configure --prefix="" --enable-zlib
```

Descrição das opções de configuração:

`--enable-zlib`
Permite que o pacote das Module-Init-Tools manipule os módulos comprimidos do kernel.

Compile o pacote:

```
make
```

Para testar os resultados, use: **`make check`**.

Instale o pacote:

```
make install
```

6.50.2. Conteúdo do ModuleModule-Init-Tools

Programas instalados: depmod, insmod, insmod.static, lsmod (vínculo para insmod), modinfo, modprobe (vínculo to insmod) e rmmod (vínculo para insmod)

Descrição rápida

depmod	Cria um arquivo de dependências baseado nos símbolos que encontra no conjunto existente de módulos; este arquivo de dependências é usado pelo **modprobe** para carregar automaticamente os módulos exigidos
insmod	Instala um módulo carregável no kernel em execução
insmod.static	Uma versão estaticamente compilada do **insmod**

lsmod	Exibe informação sobre todos os módulos carregados
modinfo	Examina um arquivo-objeto associado a um módulo do kernel e exibe todas as informações disponíveis
modprobe	Usa um arquivo de dependência tipo Makefile, criado por **depmod**, para carregar automaticamente o(s) módulo(s) relevante(s) de um conjunto de módulos disponíveis em estruturas de diretório pré-definidas
rmmod	Remove módulos carregáveis do kernel em execução

6.51. Patch-2.5.4

O pacote Patch contém um programa para modificar ou criar arquivos aplicando um arquivo “patch” (remendo) especialmente criado pelo programa **diff**.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 1.5 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make e Sed

6.51.1. Instalação do Patch

Prepare o Patch para a compilação. O parâmetro de preprocessamento *-D_GNU_SOURCE* somente é necessário em plataformas PowerPC. Pode ser removida em outras arquiteturas:

```
CPPFLAGS=-D_GNU_SOURCE ./configure --prefix=/usr
```

Compile o pacote:

```
make
```

Este pacote não vem com um conjunto de testes.

Instale o pacote:

```
make install
```

6.51.2. Conteúdo do Patch

Programa instalado: patch

Descrição rápida

patch Modifica arquivos de acordo com um patch. Uma arquivo patch é normalmente uma lista das diferenças criadas pelo programa **diff**. Aplicando estas diferenças aos arquivos originais, o **patch** cria as versões corrigidas.

6.52. Procps-3.2.5

O pacote Procps contém programas para monitorar processos.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 2.3 MB

Requisitos de instalação: Bash, Binutils, Coreutils, GCC, Glibc, Make e Ncurses

6.52.1. Instalação do Procps

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

6.52.2. Conteúdo do Procps

Programas instalados: free, kill, pgrep, pkill, pmap, ps, skill, snice, sysctl, tload, top, uptime, vmstat, w e watch
Biblioteca instalada: libproc.so

Descrição rápida

free Exibe a quantidade total de memória física e swap usada e livre no sistema, bem como a memória compartilhada e os buffers usados pelo kernel

kill Envia sinais para processos

pgrep Procura processos baseado no nome e outros atributos

pkill Sinaliza processos baseado no nome e outros atributos

pmap Retrata o mapeamento da memória de um processo

ps Exibe a situação dos processos em execução

skill Sinaliza processos que coincidem com um determinado critério

snice Modifica a prioridade de execução para processos que coincidem com um critério

sysctl Modifica os parâmetros do kernel em tempo de execução

tload Exibe um gráfico do nível de utilização do sistema para o tty especificado ou, se nenhum for fornecido, para o tty do processo do tload

top Mostra uma lista dos processos mais intensos da CPU; isto fornece uma estimativa da atividade do processador central em tempo real

uptime Relata a quanto tempo o sistema está ligado, quantos usuários estão registrados

vmstat Mostra estatísticas da memória virtual, fornecendo informações sobre processos, memória,

	paginação, input/output (IO), traps, e a atividade do processador central
w	Exibe informações sobre os usuários e seus processos atuais na máquina
watch	Executa um comando repetidamente, exibindo sua saída
`libproc`	Contém as funções usadas pela maioria dos programas deste pacote

6.53. Psmisc-21.6

O pacote Psmisc contém programas que fornecem informações sobre processos em execução.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 1.7 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses e Sed

6.53.1. Instalação do Psmisc

Prepare o Psmisc para a compilação:

```
./configure --prefix=/usr --exec-prefix=""
```

Descrição das opções de configuração:

--exec-prefix=""
Assegura que os binários do Psmisc sejam instalados em `/bin` em vez de `/usr/bin`. Esta é a posição correta de acordo com o FHS, porque alguns binários do Psmisc são usados pelo pacote LFS-Bootscripts.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Não há nenhuma razão para os programas **pstree** e **pstree.x11** ficarem em `/bin`. Mova-os para `/usr/bin`:

```
mv /bin/pstree* /usr/bin
```

Por padrão, o programa **pidof** do Psmisc's não é instalado. Isto geralmente não é um problema porque ele é instalado mais tarde com o pacote Sysvinit, que fornece um programa **pidof** melhor. Se o Sysvinit não for usado em um sistema particular, termine a instalação do Psmisc criando a seguinte ligação simbólica:

```
ln -s killall /bin/pidof
```

6.53.2. Conteúdo do Psmisc

Programas instalados: fuser, killall, pstree e pstree.x11 (vínculo para pstree)

Descrição rápida

fuser	Mostra os IDs dos processos (PIDs) que usam arquivos de dados ou arquivos do sistema
killall	Derruba processos pelo nome; emite um sinal a todos os processos em execução
pstree	Exibe os processos em execução como uma árvore
pstree.x11	O mesmos que o **pstree** exceto que espera por uma confirmação antes de sair

6.54. Shadow-4.0.9

O pacote da Shadow contém programas para manipulação de senhas em modo seguro.

Tempo de compilação aproximado: 0.4 SBU
Espaço em disco necessário: 13.7 MB

Requisitos de instalação: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Sed

6.54.1. Instalação do Shadow

Prepare o Shadow para a compilação:

```
./configure --libdir=/lib --enable-shared
```

Remova da instalação o programa **groups** e sua página man, pois o Coreutils fornece uma versão melhor:

```
sed -i 's/groups$(EXEEXT) //' src/Makefile
sed -i '/groups/d' man/Makefile
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

O Shadow utiliza dois arquivos para configurar os parâmetros de autenticação para o sistema. Instale estes dos arquivos de configuração:

```
cp etc/{limits,login.access} /etc
```

Em vez de usar o método *crypt* padrão, use o método *MD5*, mais seguro, para criptografar suas senhas, que também permite senhas maiores que 8 caracteres. É necessário mudar a posição das caixas postais do usuário, da obsoleta `/var/spool/mail` para onde o Shadow reconhece como padrão, ou seja o diretório `/var/mail` usado atualmente. Ambos podem ser realizados modificando o arquivo de configuração pertinente enquanto o copia para o seu destino:

```
sed -e's@#MD5_CRYPT_ENAB.no@MD5_CRYPT_ENAB yes@' \
    -e 's@/var/spool/mail@/var/mail@' \
    etc/login.defs.linux > /etc/login.defs
```

Mova um programa colocado em lugar errado para sua posição apropriada:

```
mv /usr/bin/passwd /bin
```

Mova as bibliotecas do Shadow para uma posição mais apropriada:

```
mv /lib/libshadow.*a /usr/lib
rm /lib/libshadow.so
ln -sf ../../lib/libshadow.so.0 /usr/lib/libshadow.so
```

A opção `-D` do programa **useradd** requer o diretório `/etc/default` para trabalhar corretamente:

```
mkdir /etc/default
```

6.54.2. Configuração do Shadow

Este pacote contém utilitários para adicionar, modificar, e suprimir usuários e grupos; define e modifica suas senhas; e executa outras tarefas administrativas. Para uma explanação detalhada do que *password shadowing* significa, veja o arquivo `doc/HOWTO` dentro da árvore desembalada dos fontes. Se for usar o suporte ao Shadow, tenha na mente que os programas que necessitam de verificação de senhas (display managers, programas de ftp, pop3, etc..) devem ser compatíveis com o shadow. Isto é, precisam poder trabalhar com senhas Shadow.

Para permitir a proteção de senhas, execute o seguinte comando:

```
pwconv
```

Permitir a proteção de senhas de grupo, execute:

```
grpconv
```

Em circunstâncias normais, as senhas não terão sido criadas ainda. Entretanto, se retornar a esta seção mais tarde para habilitar o Shadow, redefina todas as senhas atuais de usuários com o comando **passwd** ou todas as senhas de grupo com o comando **gpasswd**.

6.54.3. Definindo a senha do root

Escolha uma senha para o usuário *root* a registre com o seguinte comando:

```
passwd root
```

6.54.4. Conteúdo do Shadow

Programas instalados: chage, chfn, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmod, groups, grpck, grpconv, grpunconv, lastlog, login, logoutd, mkpasswd, newgrp, newusers, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), useradd, userdel, usermod, vigr (vínculo com vipw) e vipw
Bibliotecas instaladas: libshadow.[a,so]

Descrição rápida

chage Usado para alterar o número máximo de dias entre as mudanças obrigatórias de senha

chfn Usado para alterar o nome completo do usuário e outras informações (número da sala do escritório, número do telefone comercial e residencial).

chpasswd Usado para atualizar as senhas de um conjunto de usuários de uma vez. Lê um arquivo de nomes de usuário e senhas da entrada padrão e usa esta informação para atualizar um grupo de usuários.

chsh Usado para alterar o shell de login do usuário

expiry Verifica e reforça a política de expiração de senha

faillog	Usado para examinar o registro de falhas no login (/var/log/faillog), definir um número máximo de falhas antes que um usuário esteja bloqueado, ou zerar a contagem de falhas.
gpasswd	Usado para adicionar e excluir membros e administradores aos grupos
groupadd	Cria um grupo com os valores especificados na linha de comando e os valores-padrão do sistema
groupdel	Exclui o grupo com o nome fornecido
groupmod	Usado para alterar o nome ou o GID do grupo especificado na linha de comandoda
groups	Exibe os grupos dos quais um usuário faz parte
grpck	Verifica a integridade das informações de autenticação do sistema `/etc/group` e `/etc/gshadow`
grpconv	Converte para shadow os arquivos de grupos normais
grpunconv	Atualiza o `/etc/group` a aprtir do `/etc/gshadow` e exclui o último
lastlog	Reporta o login mais recente de todos os usuários ou de um determinado usuário
login	É usado para estabelecer uma nova sessão com o sistema
logoutd	Força o tempo de login e restrições de porta especificados em /etc/porttime
mkpasswd	Gera senhas aleatórias
newgrp	É usado para modificar o GID atual durante uma sessão de login
newusers	Lê um arquivo de nomes de usuário e senhas em texto puro e usa esta informação para atualizar um grupo de usuários existentes ou para criar novos usuários
passwd	Modifica senhas de contas de usuário e de grupos
pwck	Verifica a integridade dos arquivos de senha `/etc/passwd` e `/etc/shadow`
pwconv	Converte o arquivo de senhas normal para o arquivo shadow
pwunconv	Atualiza o `/etc/passwd` a partir do `/etc/shadow` e exclui o último
sg	Modifica o GID do usuário para o do grupo especificado, ou executa um comando dado como membro do grupo informado na linha de comando
su	Executa um shell com outros IDs de usuário e grupo
useradd	Cria um novo usuário ou atualiza informações-padrão para novos usuários
userdel	Modifica os arquivos de conta do sistema, removendo todas as entradas referentes a um nome de login especificado
usermod	Usado para modificar o nome de login do usuário informado na linha de comando, seu número de identificação de usuário (UID), o shell padrão, o grupo inicial, o diretório home, etc.
vigr	Edita os arquivos `/etc/group` ou `/etc/gshadow`
vipw	Edita os arquivos `/etc/passwd` ou `/etc/shadow`

`libshadow` Contém as funções usadas pela maioria dos programas deste pacote

6.55. Sysklogd-1.4.1

O pacote Sysklogd contém programas para gravação das mensagens de log do sistema, como aquelas reportadas pelo kernel quando coisas incomuns acontecem.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 704 KB

Requisitos de instalação: Binutils, Coreutils, GCC, Glibc e Make

6.55.1. Instalação do Sysklogd

O seguinte patch repara vários problemas, incluindo um problema na configuração do Sysklogd com kernel Linux da série 2.6

```
patch -Np1 -i ../sysklogd-1.4.1-fixes-1.patch
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

6.55.2. Configuração do Sysklogd

Crie um arquivo `/etc/syslog.conf` executando o seguinte comando:

```
cat > /etc/syslog.conf << "EOF"
# Begin /etc/syslog.conf

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

# log the bootscript output:
local2.* -/var/log/boot.log

# End /etc/syslog.conf
EOF
```

6.55.3. Conteúdo do Sysklogd

Programas instalados: klogd e syslogd

Descrição rápida

klogd Um serviço de sistema (daemon) que intercepta e registra mensagens do kernel Linux

syslogd Provê um tipo de registro de log que muitos programas modernos usam. Toda mensagem registrada contém ao menos um horário e um nome de computador e, normalmente, o nome do programa também. Mas isto depende de quão confiável é o programa sendo logado

6.56. Sysvinit-2.86

O pacote Sysvinit contém programas para controlar a inicialização, a execução e a finalização do sistema.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 1012 KB

Requisitos de instalação: Binutils, Coreutils, GCC, Glibc e Make

6.56.1. Instalação do Sysvinit

Quando os níveis de execução (run-levels) são modificados (por exemplo, quando desligando o sistema), o **init** envia os sinais de terminação TERM e KILL aos processos que ele iniciou e àqueles processos que não devem funcionar no novo nível de execução. Ao fazer isto, o **init** mostra na tela uma mensagens tais como “Sending processes the TERM signal”, o que parece dizer que está emitindo estes sinais para todos os processos atualmente em execução. Para evitar esta interpretação errônea, modifique os fontes de modo que esta mensagem seja modificada para “Sending processes started by init the TERM signal”:

```
sed -i 's@Sending processes@& started by init@g' \
    src/init.c
```

Compile o pacote:

```
make -C src
```

Instale o pacote:

```
make -C src install
```

6.56.2. Configurando o Sysvinit

Crie um novo arquivo `/etc/inittab` executando os seguintes comandos:

```
cat > /etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc sysinit

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty -I '\033(K' tty1 9600
2:2345:respawn:/sbin/agetty -I '\033(K' tty2 9600
3:2345:respawn:/sbin/agetty -I '\033(K' tty3 9600
4:2345:respawn:/sbin/agetty -I '\033(K' tty4 9600
5:2345:respawn:/sbin/agetty -I '\033(K' tty5 9600
6:2345:respawn:/sbin/agetty -I '\033(K' tty6 9600

# End /etc/inittab
EOF
```

A opção *-I '\033(K'* diz ao **agetty** para enviar esta seqüência de escape ao terminal antes de fazer qualquer outra coisa. Esta seqüência do escape modifica o conjunto de caracteres do console para que se ajuste ao definido pelo usuário, que pode ser modificado pelo programa **setfont**. O script de inicialização **console** do pacote LFS-Bootscripts chama o programa **setfont** durante a inicialização do sistema. Emitir esta seqüência de escape é necessário para quem usa fontes de tela não-ISO 8859-1, mas não afeta os usuários do inglês nativo.

6.56.3. Conteúdo do Sysvinit

Programas instalados: halt, init, killall5, last, lastb (vínculo to last), mesg, pidof (vínculo to killall5), poweroff (link to halt), reboot (link to halt), runlevel, shutdown, sulogin, telinit (link to init), utmpdump e wall

Descrição rápida

halt	Normalmente invoca o comando **shutdown** com a opção `-h`, exceto quando já no nível de execução (run-level) 0, e então informa ao kernel para suspender o sistema; grava no `/var/log/wtmp` que o sistema está sendo desligado
init	O primeiro processo a ser iniciado depois que o kernel inicializou o hardware o qual toma controle do processo de inicialização e inicia todos os processos aos quais ele está instruído a
killall5	Envia um sinal a todos os processos, exceto aos processos na sua própria seção a fim de não matar o shell que o iniciou
last	Procura através do arquivo `/var/log/wtmp` (ou do arquivo designado pela opção -f) e exibe uma lista de todos os usuários logados (ou não) desde quando aquele arquivo foi criado
lastb	Faz o mesmo que last, exceto que por padrão ele mostra um log do arquivo `/var/log/btmp` , que contém todas as tentativas falhas de login
mesg	Controla o acesso ao terminal do usuário por outras pessoas. Ele é tipicamente usado para habilitar ou desabilitar a escrita por outros usuários em seu terminal
mountpoint	Verifica se o diretório é um ponto de montagem (mountpoint)
pidof	Exibe os identificadores de processo (PIDs) dos programas citados
poweroff	Ele desativa e desliga o computador (quando usando uma BIOS com APM e o APM está habilitado no kernel) (veja **halt**)
reboot	Ele reinicia o computador (veja **halt**)
runlevel	Lê o arquivo utmp do sistema (geralmente `/var/run/utmp`), localiza o registro do nível de execução e exibe o nível de execução anterior e atual do sistema na saída padrão, separados por um espaço.
shutdown	Desativa o sistema de um modo seguro. Todos os usuários logados são notificados de que o sistema será desativado e novos logins são bloqueados
sulogin	Permite um login como *root*; é invocado pelo **init** quando o sistema entra em modo monousuário
telinit	Envia os sinais apropriados ao **init**, dizendo a ele qual nível de execução entrar
utmpdump	Exibe o conteúdo de um arquivo de login na saída padrão em um formato amigável
wall	Envia uma mensagem aos usuários logados que possuem sua permissão mesg configurada

6.57. Tar-1.15.1

O pacote Tar contém um programa de empacotamento de arquivos (archiving)

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 12.7 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make e Sed

6.57.1. Instalação do Tar

O Tar tem um erro quando a opção *-S* é usada com arquivos maiores que 4 GB. O seguinte patch repara este problema:

```
patch -Np1 -i ../tar-1.15.1-sparse_fix-1.patch
```

Prepare o Tar para a compilação:

```
./configure --prefix=/usr --bindir=/bin --libexecdir=/usr/sbin
```

Compile o pacote:

```
make
```

Para testar os resultados, use: **make check**.

Instale o pacote:

```
make install
```

6.57.2. Conteúdo do Tar

Programas instalados: rmt e tar

Descrção rápida

rmt Usado por programas de backup remoto para manipular um drive de fita magnética através de uma conexão interprocessada

tar Cria, extrai arquivos de, e lista o conteúdo dos pacotes de arquivos, conhecidos também como tarballs

6.58. Udev-056

O pacote Udev contém programas para a criação dinâmica de nós de dispositivo (device nodes).

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 6.7 MB

Requisitos de instalação: Coreutils e Make

6.58.1. Instalação do Udev

Compile o pacote:

```
make udevdir=/dev
```

udevdir=/dev
Diz ao **udev** em que diretório os nós dos dispositivos devem ser criados.

Para testar os resultados, use: **make test**.

Instale o pacote:

```
make udevdir=/dev install
```

As configurações padrão do Udev estão longe do ideal, assim instale estes arquivos de configuração:

```
cp ../udev-config-3.rules /etc/udev/rules.d/25-lfs.rules
```

Execute o programa **udevstart** para criar nosso conjunto completo de nós de dispositivos.

```
/sbin/udevstart
```

6.58.2. Conteúdo do Udev

Programas instalados: udev, udevd, udevsend, udevstart, udevinfo e udevtest
Diretório instalado: /etc/udev

Descrição rápida

udev Cria nós de dispositivos no diretório `/dev` u renomeia interfaces de rede (não no LFS) em resposta a eventos hotplug

udevd Um serviço do sistema (daemon) que reordena eventos hotplug antes de submeter ao **udev**, evitando assim saídas múltiplas

udevsend Envia eventos hotplug ao **udevd**

udevstart Cria nós de dispositivos no diretório `/dev` que correspondam aos drivers compilados diretamente no kernel; executa esta tarefa simulando eventos hotplug que presumivelmente foram emitidos pelo kernel antes da execução deste programa (por exemplo, porque o sistema de arquivos root não havia sido montado ainda), submetendo então estes eventos hotplug

simulados ao **udev**

udevinfo Permite que os usuários consultem na base de dados do **udev** as informações sobre qualquer dispositivo atualmente presente no sistema; fornece também uma maneira de consultar todos os dispositivo na árvore dos `sysfs` para ajudar a criar regras

udevtest Simula uma execução do **udev** para um dispositivo fornecido na linha de comando, e exibe o nome do nó de dispositivo que o **udev** real criaria ou o novo nome da interface de rede (não no LFS)

`/etc/udev` Este diretório contém os arquivos de configuração do **udev**, as permissões dos dispositivos, e regras para dar nomes aos dispositivos

6.59. Util-linux-2.12q

O pacote Util-linux contém programas diversos. Os mais importantes são usados para montar, desmontar, formatar, particionar e gerenciar discos rígidos, abrir portas tty e capturar mensagens do kernel.

Tempo de compilação aproximado: 0.2 SBU
Espaço em disco necessário: 11.6 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed e Zlib

6.59.1. Notas sobre a compatibilidade com o padrão FHS

O padrão FHS recomenda usar o diretório `/var/lib/hwclock` em vez do diretório usual `/etc` para o armazenamento do arquivo `adjtime`. Para fazer o programa **hwclock** compativel com o FHS, execute o que segue:

```
sed -i 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
    hwclock/hwclock.c
mkdir -p /var/lib/hwclock
```

6.59.2. Instalação do Util-linux

O Util-linux falha ao compilar com as versões mais novas do Linux-Libc-Headers. O seguinte patch elimina este problema:

```
patch -Np1 -i ../util-linux-2.12q-cramfs-1.patch
```

Prepare o Util-linux para a compilação:

```
./configure
```

Compile o pacote:

```
make HAVE_KILL=yes HAVE_SLN=yes
```

Descrição das opções de configuração:

HAVE_KILL=yes
Impede que o programa **kill** (já instalado pelo Procps) seja instalado outra vez.

HAVE_SLN=yes
Impede que o programa **sln** uma versão estaticamente vinculada do **ln** já instalado pelo Glibc) seja instalado outra vez.

Este pacote não vem com um conjunto de testes.

Instale o pacote e mova o binário **logger** para o diretório `/bin` ser uma exigência do pacote LFS-Bootscripts:

```
make HAVE_KILL=yes HAVE_SLN=yes install
mv /usr/bin/logger /bin
```

6.59.3. Conteúdo do Util-linux

Programas instalados: agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, fsck.cramfs, fsck.minix, getopt, hexdump, hwclock, ipcrm, ipcs, isosize, line, logger, look, losetup, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, namei, pg, pivot_root, ramsize (vínculo to rdev), raw, rdev, readprofile, rename, renice, rev, rootflags (vínculo to rdev), script, setfdprm, setsid, setterm, sfdisk, swapdev, swapoff (vínculo to swapon), swapon, tunelp, ul, umount, vidmode (vínculo to rdev), whereis e write

Descrição rápida

agetty	Abre uma porta tty, pede um nome de login e invoca o comando **login**
arch	Exibe a arquitetura da máquina
blockdev	Permite chamar dispositivos de controle de I/O de bloco (block device ioctls) na linha de comando
cal	Exibe um calendário simples
cfdisk	Manipulador da tabela de partição baseado no libncurses
chkdupexe	Procura executáveis duplicados
col	Filtra LFs (line feeds) reversos da entrada
colcrt	Filtra a saída do **nroff** para visualização em CRT
colrm	Remove colunas de um arquivo
column	Formata um arquivo em múltiplas colunas
ctrlaltdel	Configura a função da combinação de teclas CTRL+ALT+DEL (desligamento soft ou hard)
cytune	Pesquisa e modifica a interrupção para o driver Cyclades
ddate	Converte datas Gregorianas para Discordianas
dmesg	Usado para examinar ou controlar o buffer de mensagens do kernel (mensagens de inicialização).
elvtune	Permite a você ajustar o elevador I/O com base na fila de dispositivos de blocos.
fdformat	Formata um disquete em baixo nível
fdisk	Manipulador da tabela de partição do disco
fsck.cramfs	Executa uma verificação de consistência no sistema de arquivos Cramfs no dispositivo especificado na linha de comando
fsck.minix	Executa uma verificação de consistência no sistema de arquivos minix no dispositivo especificado na linha de comando
getopt	Analisa opções de comandos da mesma forma que a função getopt do C.
hexdump	Exibe os arquivos especificados, ou a entrada padrão, em um dado formato (ascii, decimal, hexadecimal, octal).

hwclock Exibe e configura o relógio de hardware (também chamado de RTC ou relógio da BIOS).

ipcrm Remove um recurso especificado de uma comunicação inter-Process (IPC).

ipcs Fornece a informação de status do IPC

isosize Exibe o tamanho de um sistema de arquivos iso9660

line Copia uma linha (até a nova-linha) da entrada padrão e a escreve na saída padrão

logger Cria entradas no log do sistema

look Exibe linhas que começam com uma dada string

losetup Configura e controla dispositivos loop

mcookie Gera cookies mágicos (números 128-bit hexadecimais aleatórios) para o **xauth**

mkfs Cria um sistema de arquivos Linux em um dispositivo, geralmente uma partição do disco

mkfs.bfs Cria um sistema de arquivos SCO bfs em um dispositivo, geralmente uma partição do di

mkfs.cramfs Cria um sistema de arquivos cramfs em um dispositivo, geralmente uma partição do disco

mkfs.minix Cria um sistema de arquivos MINIX em um dispositivo, geralmente uma partição do disco

mkswap Configura a memória swap em uma partição ou arquivo

more Um filtro para paginação de textos em tela cheia

mount Monta, de várias fontes possíveis, sistemas de arquivos ou diretórios em um diretório (ponto de montagem)

namei Percorre um caminho de arquivo até um ponto terminal ser encontrado

pg Mostra um arquivo texto uma tela por vez

pivot_root Modifica o sistema de arquivos raiz do processo atual

ramsize Exibe e configura o tamanho do disco RAM

raw Usado para ligar um dispositivo de caractere cru a um dispositivo de bloco

rdev Exibe e configura o dispositivo raiz de uma imagem, dispositivo swap, disco RAM, ou modo de vídeo

readprofile Lê informações de perfil do kernel

rename Renomeia arquivos

renice Altera a prioridade de processos em execução

rev Inverte as linhas de um arquivo

rootflags Exibe e configura informações extras usadas ao montar o diretório raiz

script Cria uma impressão da sessão de terminal

setfdprm Configura parâmetros do disquete fornecidos pelo usuário

setsid Executa programas em uma nova sessão

setterm Configura os atributos do terminal

sfdisk Manipulador da tabela de partição do disco

swapdev Ajusta dispositivos e arquivos para paginação e armazenamento temporário

swapoff Desabilita dispositivos e arquivos para paginação e armazenamento temporário

swapon Habilita dispositivos e arquivos para paginação e armazenamento temporário

tunelp Configura vários parâmetros para o dispositivo LP

ul Um filtro que lê um arquivo e traduz ocorrências de underscores para a seqüência que indica sublinhado no terminal em uso

umount Desmonta um sistema de arquivos ou diretório

vidmode Exibe e configura o modo de vídeo

whereis Localiza o binário, arquivo-fonte e página de manual de um comando

write Envia uma mensagem para outro usuário, *se* aquele usuário tem escrita habilitada

6.60. Sobre os símbolos de debbug

A maioria dos programas e das bibliotecas, por padrão, são compilados com os símbolos de debbug incluídos (com a opção `-g` do **gcc**). Isto significa que ao eliminar erros de um programa ou biblioteca que foi compilado com estes símbolos, o debugger pode fornecer não somente os endereços de memória, mas também os nomes das rotinas e das variáveis.

Entretanto, a inclusão destes símbolos de debug aumenta o tamanho do programa ou da biblioteca significativamente. Veja um exemplo da quantidade de espaço que estas marcas ocupam:

- Binário bash com símbolos de debbug: 1200 KB
- Binário bash sem símbolos de debbug: 480 KB
- Glibc e arquivos do GCC (`/lib` e `/usr/lib`) com símbolos de debbug: 87 MB
- Glibc e arquivos do GCC sem símbolos de debbug: 16 MB

Os tamanhos podem variar dependendo do compilador e da biblioteca C utilizados, mas ao comparar programas com e sem símbolos de debbug, a diferença será geralmente um fator entre dois e cinco.

Como a maioria de usuários nunca usarão um debugger, muito espaço em disco pode ser recuperado removendo estes símbolos. A seção seguinte mostra como remover todos os símbolos de debug dos programas e das bibliotecas. Informações adicionais de otimização do sistema podem ser encontradas em *http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt*.

6.61. Stripping novamente

Se o usuário não for um programador e não pretende fazer debbug no software de seu sistema, o tamanho do sistema pode diminuir aproximadamente 200 MB removendo os símbolos de debbug dos binários e das bibliotecas. Isto não causa nenhum inconveniente, exceto não poder fazer o debbug dos programas.

A maioria das pessoas que usam o comando abaixo não têm nenhuma dificuldade. Entretanto, é fácil cometer um erro e tornar o sistema novo inutilizável. Assim, antes de executar o comando **strip**, é uma idéia fazer um backup do sistema atual.

Antes de executa-lo tome cuidado especial em verificar se nenhuns dos binários esteja em execução. Se não tiver certeza do usuário incorporado no comando chroot usado em Section 6.3, “Entrando no ambiente Chroot,”, primeiro saia do chroot:

```
logout
```

Reentre então com:

```
chroot $LFS /tools/bin/env -i \
    HOME=/root TERM=$TERM PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin \
    /tools/bin/bash --login
```

Agora podemos aplicar o strip nos binários e nas bibliotecas com segurança:

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
   -exec /tools/bin/strip --strip-debug '{}' ';'
```

Um grande número de arquivos serão reportados como tendo formato não reconhecido. Estes avisos podem com segurança ser ignorados, pois indicam apenas que aqueles arquivos são scripts, e não binários.

Se o espaço de disco estiver muito apertado, a opção `--strip-all` ser utilizada nos binários dos diretórios `/{,usr/}{bin,sbin}` para ganhar ainda mais megabytes. Não use esta opção em bibliotecas—elas serão destruídas.

6.62. Últimos cuidados

De agora em diante, ao reentrar no ambiente chroot depois de sair por qualquer motivo, use o comando chroot da seguinte forma:

```
chroot "$LFS" /usr/bin/env -i \
    HOME=/root TERM="$TERM" PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin \
    /bin/bash --login
```

A razão para isto é que os programas em `/tools` não são mais necessários. Então você pode até apagar este diretório, ou empacotá-lo (com o tar) se pretende configurar um outro sistema lfs.

Note

Remover o diretório `/tools` também removerá as cópias provisórias do Tcl, Expect, e DejaGNU que foram usadas durante a execução dos testes do conjunto de ferramentas (toolchain). Se você precisar destes programas mais tarde, eles terão que ser recompilados e reinstalados. O livro BLFS tem instruções sobre isto (veja em *http://www.linuxfromscratch.org/blfs/*).

Chapter 7. Configurando os scripts de inicialização do sistema

7.1. Introdução

Este capítulo detalha como instalar e configurar o pacote LFS-Bootscripts. A maioria destes scripts trabalharão sem modificações, mas alguns vão exigir arquivos adicionais de configuração porque tratam de informações que dependem do hardware.

Os scripts de inicialização ao estilo System-V são utilizados neste livro porque são largamente aceitos. Para opções adicionais, uma sugestão de leitura que detalha a instalação dos scripts de inicialização ao estilo do DEB está disponível em *http://www.linuxfromscratch.org/hints/ downloads/files/bsd-init.txt*. Procurando nas listas de discussão do LFS por #depinit# você vai encontrar outras sugestões alternativas.

Se for utilizar um estilo alternativo nos scripts de inicialização, pule este capítulo e vá direto para o Chapter 8.

7.2. LFS-Bootscripts-3.2.1

O pacote LFS-Bootscripts contém um conjunto de scripts de inicialização/finalização do sistema LFS durante o bootup/shutdown.

Tempo de compilação aproximado: 0.1 SBU
Espaço em disco necessário: 0.3 MB

Requisitos de instalação: Bash e Coreutils

7.2.1. Instalação do LFS-Bootscripts

Instale o pacote:

```
make install
```

7.2.2. Conteúdo do LFS-Bootscripts

Scripts instalados: checkfs, cleanfs, console, functions, halt, hotplug, ifdown, ifup, localnet, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysklogd, template e udev

Descrição rápida

checkfs Verifica a integridade dos sistemas de arquivos antes que sejam montados (exceto os sistemas de arquivos baseados em journal e rede).

cleanfs Remove os arquivos que não devem ser preservados entre as reinicializações, tais como aqueles em `/var/run/` e `/var/lock/`; recria o `/var/run/utmp` e remove os arquivos `/etc/nologin`, `/fastboot` e `/forcefsck`, quando presentes

console Carrega a tabela correta do keymap para o layout de teclado desejado; ajusta também a fonte de tela

functions Contém as funções comuns, como as de verificação de erro e status, que são usadas por diversos scripts

halt Desliga o sistema

hotplug Carrega módulos para dispositivos do sistema

ifdown Auxiliar ao script de rede para a finalização dos dispositivos de rede

ifup Auxiliar ao script de rede para a inicialização dos dispositivos de rede

localnet Define os dispositivos hostname e local loopback do sistema

mountfs Monta todos os sistemas de arquivos, exceto os que estão definidos como *noauto* e os baseados em rede

mountkernfs Monta os sistemas de arquivo virtuais do kernel, como o `proc`

network Configura as interfaces de rede, como a placa de rede, e define o gateway padrão (onde aplicável).

rc O script mestre de controle dos níveis de execução (run-level); é responsável pela execução de todos os demais bootscripts, um a um, em uma seqüência determinada pelo nome das ligações simbólicas que estão sendo processadas

reboot Reinicializa o sistema

sendsignals Certifica-se que cada processo está terminado antes que o sistema reinicialize ou desligue

setclock Ajusta o relógio do kernel para a hora local quando o relógio do hardware não está ajustado com a hora UTC

static Fornece a funcionalidade necessária para atribuir um endereço IP (Internet Protocol) estático para uma interface de rede

swap Habilita e desabilita os arquivos e a partição de troca (swap).

sysklogd Inicia e finaliza os log daemons do sistema e do kernel

template Um modelo para criar scripts de inicialização padronizados para outros serviços do sistema

udev Prepara o diretório `/dev` e inicializa o Udev

7.3. Como estes scripts de inicialização trabalham?

O Linux utiliza um recurso simples de inicialização, SysVinit, que é baseado no conceito de *run-levels* (níveis de execução). Pode ser completamente diferente em um sistema com relação a outro, assim não se pode supor que porque as coisas funcionam em uma distribuição particular do Linux devem trabalhar da mesma forma em um sistema LFS. O LFS tem sua própria maneira de fazer coisas, mas respeita os padrões geralmente aceitos.

O SysVinit (que será chamado de "init" de agora em diante) trabalha usando um esquema de níveis de execução. Há sete níveis (numerados de 0 a 6). Na realidade, existem outros, mas são usados apenas em casos especiais. Veja `init(8)` para mais detalhes. Cada um deles corresponde às ações que o computador está supostamente executando durante a inicialização. O nível de execução padrão é o 3. Estas são as descrições dos diferentes níveis quando são executados:

```
0: desligar o computador
1: modalidade mono-usuário
2: modalidade multi-usuário sem suporte à rede
3: modalidade multi-usuário com suporte à rede
4: reservado para configuração, no mais faz o mesmo que 3
5: mesmos que 4, é usado geralmente para o início de uma sessão do GUI (como o xdm do x ou kdm do KDE)
6: reinicilizar o computador
```

O comando utilizado para alterar o nível de execução é **init** ***`[runlevel]`*** onde *`[runlevel]`* é o nível desejado. Por exemplo, para reinicializar o computador, um usuário poderia usar o comando **init 6**, que é um alias (apelido) para o comando **reboot**. Da mesma forma, **init 0** é um apelido para o comando **halt** (pare).

Existem alguns subdiretórios em `/etc/rc.d` com o formato `rc?.d` (onde ? é o número do nível de execução) e `rcsysinit.d`, todos contendo um certo número de vínculos simbólicos. Alguns começam com um *K*, outros começam com um *S*, e todo têm dois números juntos à letra inicial. O K (do inglês, kill) significa mate (ou finalize) um serviço e o S (do inglês, start) significa inicie um serviço. Os números determinam a ordem em que os scripts são executados, de 00 a 99 — quanto menor o número, mais cedo um serviço é executado. Quando o **init** alterna para outro nível de execução, os serviços apropriados são interrompidos ou inicializados, conforme o nível escolhido.

Os scripts reais estão em `/etc/rc.d/init.d`. Eles fazem todo o trabalho e os vínculos simbólicos apontam para eles. Vínculos simbólicos de inicialização e finalização apontam para o mesmo script em `/etc/rc.d/init.d`. Isto acontece porque os scripts podem ser chamados com parâmetros diferentes, como *`start`*, *`stop`*, *`restart`*, *`reload`* e *`status`*. Quando um vínculo K é encontrado, o script apropriado é executado com a opção *`stop`*. Quando um vínculo S é encontrado, o script apropriado é executado com o parâmetro *`start`*.

Há uma exceção. Vínculos que iniciam com um *S* nos diretórios `rc0.d` e `rc6.d` não iniciarão nenhum serviço. Os scripts serão chamados com o parâmetro *`stop`* para interromper um serviço. A lógica por detrás disso é que, quando for reiniciar ou desligar o sistema, você não deseja iniciar nada, apenas interromper o sistema.

Estas são descrições das funções dos argumentos nos scripts:

`start`
: O serviço é iniciado.

stop
: O serviço é interrompido.

restart
: O serviço é interrompido e novamente iniciado.

reload
: A configuração do serviço é atualizada. É utilizado após a modificação do arquivo de configuração de um serviço, quando este não precisa ser reiniciado.

status
: Diz se o serviço está rodando e com quais PIDs.

Sinta-se à vontade para modificar o modo como o processo de inicialização trabalha (afinal, é o seu próprio sistema LFS). Os arquivos oferecidos aqui são apenas um exemplo de como isto pode ser feito.

7.4. Manipulando dispositivos e módulos em um sistema LFS

No Chapter 6, nós instalamos o pacote Udev. Antes de entrarmos nos detalhes a respeito de como ele trabalha, vamos fazer um breve histórico dos métodos de manipulação de dispositivos.

Os sistemas Linux geralmente usam o tradicional método de criação estática de dispositivos, pelo qual muitos nós de dispositivos (device nodes) são criados no diretório `/dev` (às vezes milhares de nós, literalmente), correspondentes aos dispositivos de hardware existentes no sistema. Isto é tipicamente feito pelo script **MAKEDEV**, que faz algumas chamadas ao programa **mknod** com a identificação numérica principal e secundária dos dispositivos para cada dispositivo que possa existir no mundo. Usando o método Udev, somente aqueles dispositivos que são destacados pelo kernel inicializam os nós de dispositivo criados para eles. Como estes nós de dispositivos são criados a cada inicialização do sistema, serão armazenados em um sistema de arquivos do tipo `tmpfs` (um sistema de arquivos virtual que reside inteiramente na memória do sistema). Os nós do dispositivo não exigem muito espaço, assim a memória usada é insignificante.

7.4.1. Histórico

Em fevereiro 2000, um novo sistema de arquivos chamado `devfs` foi incorporado no kernel 2.3.46 e tornado disponível na séries 2.4 de kernels estáveis. Embora estivesse presente no próprio código-fonte do kernel, este método de criar dispositivos dinamicamente nunca recebeu o suporte decisivo dos desenvolvedores do núcleo do kernel.

O problema principal com a abordagem adotada pelo `devfs` era o modo como manipulava a detecção, criação e nomeação de dispositivos. Esta última etapa, quando o nó de dispositivo recebe um nome, era talvez a mais crítica. Como os nomes dos dispositivos são passíveis de configuração, seria aceitável então que a política para dar nomes aos dispositivo fosse do administrador do sistema e não imposta por algum desenvolvedor qualquer. O sistema de arquivos `devfs` sofre também com algumas condições que são inerentes ao seu projeto e não podem ser eliminadas sem uma revisão substancial do kernel. Ficou marcado também pela desatualização devido à falta de manutenção.

Com o desenvolvimento das versões instáveis 2.5 do kernel, liberado mais tarde com a série estável 2.6 do kernel, surge um novo sistema de arquivos virtual chamado `sysfs`. O trabalho do `sysfs` é fornecer uma visão da configuração do hardrware do sistema para os processos, ao nível do usuário. Com esta nova representação (userspace-visible representation), a possibilidade de ser vista uma modificação na `devfs` ao nível do usuário tornou-se muito mais realista.

7.4.2. Implementação Udev

O sistema de arquivos `sysfs` foi rapidamente mencionado anteriormente. Mas como o `sysfs` pode saber sobre os dispositivos existentes no sistema e que identificações numéricas de dispositivos usar? Os drivers que tenham sido compilados no kernel registram diretamente seus objetos na `sysfs` assim que são detectados pelo kernel. Para os drivers compilados como módulos, este registro acontecerá quando o módulo é carregado. Assim que o sistema de arquivos `sysfs` é montado (em `/sys`), os dados que os drivers internos registraram no `sysfs` estão disponíveis para os processos ao nível de usuário e para o **udev** que cria os nós dos dispositivos.

O script de inicialização **S10udev** cria estes nós de dispositivos quando Linux é carregado. Este script começa registrando o **/sbin/udevsend** como um alimentador de eventos hotplug. Os eventos Hotplug (discutidos abaixo) normalmente não são gerados neste estágio, mas o **udev** é registrado para o caso de isto vir a acontecer. O programa **udevstart** então percorre todo o sistema de arquivos `/sys` e cria, em `/dev`, os dispositivos que combinem com as descrições. Por exemplo, `/sys/class/tty/vcs/dev` contém a string “7:0”. Esta string

é usada pelo **udevstart** para criar o `/dev/vcs` com o número maior *7* e menor *0*. Os nomes e as permissões dos nós criados sob o diretório `/dev` são configurados de acordo com as diretrizes especificadas nos arquivos dentro do diretório `/etc/udev/rules.d/`. Estes são numerados de forma similar pelo pacote LFS-Bootscripts. Se o **udev** não puder encontrar uma diretriz para o dispositivo que está criando, ele optará pela permissão *660* e pela propriedade do *root:root*.

Assim que o estágio acima estiver completo, todos os dispositivos presentes e que têm compilados seu drivers estarão disponíveis para uso. Isto nos leva aos dispositivos que têm drivers modulares.

Mais cedo, nós mencionamos o conceito de um “gerenciador de eventos do hotplug”. Quando a conexão de um novo dispositivo é detectada pelo kernel, ele gera um evento hotplug e busca pelo arquivo `/proc/sys/kernel/hotplug` para determinar qual o programa ao nível do usuário manipula a conexão do dispositivo. O **udev** está registrado pelo script de inicialização **udevsend** como sendo este gerenciador. Quando estes eventos do hotplug são gerados, o kernel dirá ao **udev** para verificar o sistema de arquivos `/sys` para ver se há informações sobre este novo dispositivo e para criar a entrada `/dev` para ele.

Isto nos leva a um problema que existe com o **udev** e que também havia com o `devfs`, antes dele. É conhecido como o problema “do ovo e da galinha”. A maioria de distribuições Linux manipulam o carregamento de módulos através das entradas do arquivo `/etc/modules.conf`. O acesso a um nó do dispositivo causa o carregamento do módulo do kernel apropriado. Com o **udev**, este método não funciona porque o nó do dispositivo não existe até que o módulo seja carregado. Para resolver isto, o script de inicialização **S05modules** foi adicionado ao pacote LFS-Bootscripts, junto com o arquivo `/etc/sysconfig/modules`. Acrescentando nomes de módulos ao arquivo `modules`, estes módulos serão carregado quando o computador inicializar. Isto permite ao **udev** detectar os dispositivos e criar os nós apropriados.

Note que em máquinas mais lentas ou para os drivers que criam muitos nós de dispositivo, o processo de configuração dos dispositivos pode fazer exames que demoram alguns segundos para terminar. Isto significa que alguns nós de dispositivos podem não estar imediatamente acessíveis.

7.4.3. Manipulando dispositivos dinâmicos (hotpluggable)

Quando você conecta (plug) um dispositivo, tal como um tocador MP3 USB, o kernel reconhece que o dispositivo foi conectado e gera um evento hotplug. Se o driver estiver carregado (ou porque foi compilado no kernel ou porque foi carregado pelo script de inicialização **S05modules**), o **udev** será chamado para criar o respectivo nó de dispositivo de acordo com os dados do `sysfs` disponíveis em `/sys`.

Se o driver para o dispositivo conectado estiver disponível como módulo, mas não estiver carregado no momento, o pacote Hotplug carregará o módulo apropriado e deixará este dispositivo disponível criando um nó para ele.

7.4.4. Problemas com a criação de dispositivos

Estes são alguns problemas comuns que podem surgir com a criação automática de nós de dispositivos:

1) Um kernel driver não pode exportar seus dados para o `sysfs`.

Isto é mais comum com drivers de terceiros não incorporados ao kernel. O Udev será incapaz de criar automaticamente nós do dispositivo para tais drivers. Use o arquivo de configuração `/etc/sysconfig/createfiles` para criar manualmente os dispositivos. Consulte o arquivo `devices.txt` que acompanha a documentação do kernel ou a documentação do driver para encontrar os números apropriados para principal/secundário.

2) Um dispositivo que não seja um hardware é necessário. Isto é comum com o projeto Advanced Linux Sound

Architecture (ALSA) do módulo de compatibilidade Open Sound System (OSS). Estes tipos de dispositivos podem ser manipulados de duas maneiras:

- Adicionando o nome do módulo ao arquivo `/etc/sysconfig/modules`

- Usando uma linha "install" no arquivo `/etc/modprobe.conf`. Isto diz ao comando **modprobe**" para quando carregar este módulo, carregar também este outro módulo, ao mesmo tempo". Por exemplo:

```
install snd-pcm modprobe -i snd-pcm ; modprobe \
    snd-pcm-oss ; true
```

  Isto fará com que o sistema carregue os dois módulos *snd-pcm* e *snd-pcm-oss* quando for requerido o carregamento do driver *snd-pcm*.

7.4.5. Leitura Útil

Uma documentação útil adicional está disponível nos seguintes locais:

- A Userspace Implementation of `devfs` *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf*
- udev FAQ *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-FAQ*
- The Linux Kernel Driver Model *http://public.planetmirror.com/pub/lca/2003/proceedings/papers/Patrick_Mochel/Patrick_Mochel.pdf*

7.5. Configurando o script setclock

O script **setclock** obtém a data/hora do relógio do computador, também conhecido como relógio do BIOS ou Complementary Metal Oxide Semiconductor (CMOS). Se o relógio do computador estiver ajustado ao UTC, este script converterá a data/hora para o horário local usando o arquivo `/etc/localtime` (que diz ao programa **hwclock** qual a timezone do usuário). Não há nenhuma maneira de se detectar se relógio do computador está ou não o ajustado ao UTC, isto precisa ser configurado manualmente.

Se você não sabe se o relógio do computador está ajustado ao UTC, descubra com o comando **`hwclock --localtime --show`** Isto mostrará a data/hora do seu compudador. Se estiver de acordo com o que o seu relógio de pulso ou de parede diz, o relógio do seu sistema está ajustado à hora local (é o mais comum em desktops). Se a saída do **hwclock** não for a hora local, é possível que seu computador esteja ajustado ao UTC. Verifique isto adicionando ou subtraindo a quantidade apropriada de horas conforme a timezone mostrada pelo **hwclock**. Por o exemplo, se você estiver atualmente no timezone MST, conhecido também como GMT -0700, adicione sete horas.

Mude o valor da variável `UTC` abaixo para *`0`* (zero) se o relógio do sistema *não* estiver ajustado ao UTC.

Crie um novo arquivo `/etc/sysconfig/clock` com o seguinte procedimento:

```
cat > /etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock

UTC=1

# End /etc/sysconfig/clock
EOF
```

Uma boa dica que explica como lidar com o tempo em um sistema LFS está disponível em *http://www.linuxfromscratch.org/ hints/downloads/files/time.txt*. Ele explica recursos como as time zones, UTC, e a variável de ambiente de `TZ`.

7.6. Configurando o terminal Linux

Esta seção discute como configurar o script de inicialização **console** que define o mapa do teclado e a fonte do terminal. Se caracteres não-ASCII (por exemplo, o sinal britânico da libra e o símbolo monetário do euro) não forem usados, e o teclado for do padrão U.S., salte esta seção. Sem um arquivo de configuração, o script de inicialização **console** não fará nada.

O script **console** lê o arquivo `/etc/sysconfig/console` para obter as informações de configuração. Saiba qual a fonte de tela e o keymap que devem ser utilizados. Consultar o HOWTO de configuração para a sua língua ajuda nesta tarefa (veja *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. Um arquivo `/etc/sysconfig/console` pré-configurado com as definições conhecidas para diversos países foi instalado com o pacote LFS-Bootscripts, assim a seção pertinente pode ser simplesmente descomentada se houver suporte para o seu país. Se ainda estiver em dúvida, olhe no diretório `/usr/share/kbd` para ver os mapas de teclas e as fontes de tela válidas. Leia `loadkeys(1)` e `setfont(8)` para saber os argumentos corretos para estes programas. Tudo decidido, crie o arquivo de configuração com o seguinte comando:

```
cat >/etc/sysconfig/console <<"EOF"
KEYMAP="[arguments for loadkeys]"
FONT="[arguments for setfont]"
EOF
```

Por exemplo, para os usuários espanhóis que querem também usar o caracter monetário do euro (teclando AltGr+E), os seguintes ajustes são os indicados:

```
cat >/etc/sysconfig/console <<"EOF"
KEYMAP="es euro2"
FONT="lat9-16 -u iso01"
EOF
```

Note

A linha `FONT` acima está definida para o uso do conjunto de caracteres do ISO 8859-15. Para usar o ISO 8859-1 e, conseqüentemente, o sinal da libra em vez de euro, a linha correta para `FONT` seria:

```
FONT="lat1-16"
```

Se a variável `KEYMAP` ou `FONT` não estiver definida, o script de inicialização **console** não executará os programas correspondentes.

Em alguns mapas de teclado, as teclas de retrocesso (Backspace) e de supressão (Delete) emitem sinais diferentes do keymap padrão configurado no kernel. Isto confunde algumas aplicações. Por exemplo, o Emacs exibe o arquivo de ajuda (em vez de apagar o caracter antes do cursor) quando o retrocesso é pressionado. Verificar se é caso do keymap em uso no seu sistema (isto funciona somente para os keymaps i386):

```
zgrep '\W14\W' [/path/to/your/keymap]
```

Se o keycode 14 for Backspace em vez de Delete, crie o seguinte snippet do keymap para solucionar isto:

```
mkdir -p /etc/kbd && cat > /etc/kbd/bs-sends-del <<"EOF"
                  keycode  14 = Delete Delete Delete Delete
              alt keycode  14 = Meta_Delete
        altgr alt keycode  14 = Meta_Delete
                  keycode 111 = Remove
    altgr control keycode 111 = Boot
      control alt keycode 111 = Boot
altgr control alt keycode 111 = Boot
EOF
```

Diga ao script **console** para carregar este snippet após o keymap principal:

```
cat >>/etc/sysconfig/console <<"EOF"
KEYMAP_CORRECTIONS="/etc/kbd/bs-sends-del"
EOF
```

Para compilar o keymap diretamente no kernel em vez de ajustá-la todas as vezes pelo script de inicialização **console**, siga as instruções dadas na Section 8.3, "Linux-2.6.11.12.". Fazer isto assegura que o teclado funcionará sempre como esperado, mesmo quando o sistema for inicializado no modo manutenção (passando a opção *init=/bin/sh* para o kernel), porque o script de inicialização **console** não é executado nesta situação. Por outro lado, o kernel não definirá a fonte de tela automaticamente. Isto não deve ser um grande problema porque os caracteres ASCII serão manipulados corretamente e é improvável que um usuário necessite de caracteres não-ASCII quando no modo de manutenção.

Como neste caso o kernel definirá o keymap, é possível omitir a variável KEYMAP do arquivo de configuração /etc/sysconfig/console. Pode também ser deixado lá, sem nenhuma conseqüência. Mantê-lo poderia ser benéfico por exemplo quando se executam diversas versões ou compilações do kernel em uma mesmo equipamento, e fique difícil ter certeza de que o keymap esteja compilado em cada delas.

7.7. Configurando o script sysklogd

O script `sysklogd` chama o programa **syslogd** com a opção *`-m 0`*. Esta opção desativa a marcação periódica de tempo (periodic timestamp mark) do **syslogd** que faz os registros nos arquivos de log a cada 20 minutos. Se você quiser habilitar este modo, edite o script `sysklogd` e faça as alterações necessárias. Veja a página **`man syslogd`** para mais informações.

7.8. Criando o arquivo /etc/inputrc

O arquivo `inputrc` modifica o mapa do teclado para situações específicas. Ele é o arquivo de inicialização do Readline — biblioteca relacionada ao input — utilizado pelo Bash e pela maioria dos pacotes shell.

A maioria das pessoas não necessita de um mapeamento de teclado específico para o usuário, assim, o comando abaixo cria um `/etc/inputrc` global utilizado por todos que façam logon no sistema. Se você mais tarde precisar substituir o padrão por uma base por usuário, você pode criar um arquivo `.inputrc` no diretório home do usuário com o mapeamento modificado.

Para mais informação sobre como editar o arquivo `inputrc`, veja a seção *Readline Init File* em **info bash**. O **info readline** é também uma boa fonte de info.

Segue abaixo um `inputrc` global e genérico, junto com os comentários explicativos das várias opções. Note que os comentários não podem estar na mesma linha que os comandos. Crie o arquivo a usando o seguinte comando:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the
# value contained inside the 1st argument to the
# readline specific functions

"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
```

```
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

7.9. Os arquivos de inicialização do shell bash

O programa de shell **/bin/bash** (daqui por diante chamado de “o shell”) utiliza um conjunto de arquivos de inicialização que auxilia na criação do ambiente interativo. Cada arquivo tem um uso específico e pode afetar o login e o ambiente de trabalho de formas diferentes. Os arquivos no diretório `/etc` fornecem ajustes globais. Se um arquivo equivalente existir no diretório /home do usuário, suas definições podem se sobrepor aos ajustes globais.

Um ambiente interativo login-shell é iniciado após um login bem sucedido, usando o **/bin/login**, lendo o arquivo `/etc/passwd`. Um ambiente non-login-shell é iniciado na linha de comando (por exemplo, `[prompt]$`**/bin/bash**). Um shell não-interativo está geralmente presente quando um shellscript está sendo executado. É não-interativo porque está processando um script e não espera nenhum input do usuário entre os comandos.

Para mais informação, veja *Bash Startup Files and Interactive Shells* em **info bash**.

Os arquivos `/etc/profile` e `~/.bash_profile` são processados quando o bash é iniciado como um shell interativo no início de uma sessão (login-shell).

O `/etc/profile` básico, configurado abaixo, define algumas variáveis de ambiente necessárias para o suporte à língua nativa. Ajustá-lo corretamente resulta em:

- A saída dos programas traduzidos para a língua nativa
- Classificação correta dos caracteres em letras, em dígitos e em outras classes. Isto é necessário para o **bash** para aceitar corretamente caracteres de não-ASCII em linhas de comando em locales não-Ingleses
- A ordem de classificação alfabética correta para o país
- Tamanho do papel padrão
- Formato monetário, de tempo, e de datas

Este script ajusta também a variável de ambiente `INPUTRC` que faz o bash e o Readline utilizarem o arquivo `/etc/inputrc` criado na seção anterior.

Substitua o *`[ll]`* nos comandos abaixo pelo código de duas letras correspondente à língua desejada (por exemplo, “en” para o inglês) e *`[CC]`* com o código de duas letras correspondente ao seu país (por exemplo, “GB” para Grã-Bretanha). Substitua *`[charmap]`* pelo charmap canônico do locale escolhido.

A lista de todos os locales suportados pela Glibc pode ser obtida executando o seguinte comando:

```
locale -a
```

Os locales podem ter muitos sinônimos, por exemplo “ISO-8859-1” é também referido como “iso8859-1” e “iso88591”. Algumas aplicações não podem manipular os vários sinônimos corretamente, assim é o mais seguro escolher o nome canônico para um locale particular. Para determinar o nome canônico, execute o seguinte comando, onde *`[nome do locale]`* é a saída dada pelo **locale -a** para seu locale preferido (“en_GB.iso88591” em nosso exemplo).

```
LC_ALL=[locale name]
locale charmap
```

Para o locale “en_GB.iso88591”, o comando acima mostrará:

```
ISO-8859-1
```

Isto finalmente resulta em um locale do tipo “en_GB.ISO-8859-1”.

Uma vez que os ajustes apropriados do locale foram determinados, crie o arquivo `/etc/profile`:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

export LANG=[ll]_[CC].[charmap]
export INPUTRC=/etc/inputrc

# End /etc/profile
EOF
```

Note

O “C” (padrão) e o “en_US” (recomendado para usuários dos EUA) são locales diferentes.

Definir o mapa do teclado, a fonte de tela, e as variáveis de ambiente relacionadas ao locale são os únicos procedimentos de internacionalização necessários para suportar os locales que usam caracteres de byte simples (single-byte encodings) e o sentido esquerda-para-direita de escrita. Alguns casos mais complexos (locales baseados no Utf-8 inclusive) exigem etapas adicionais e patches adicionais porque muitas aplicações tendem a não trabalhar corretamente sob tais circunstâncias. Estas etapas e remendos não são incluídos no livro LFS e tais locales ainda não são suportados pelo LFS.

7.10. Configurando o script localnet

A configuração do nome do computador (hostname) é parte do script **localnet** Isto é feito no arquivo `/etc/sysconfig/network`.

Crie o arquivo `/etc/sysconfig/network` e dê um nome para seu computador (hostname) executando:

```
echo "HOSTNAME=[lfs]" > /etc/sysconfig/network
```

Substitua *`[lfs]`* pelo nome que o seu computador será conhecido. Você não deve colocar o FQDN (Fully Qualified Domain Name, Nome de Domínio Completamente Qualificado) aqui. Esta informação será colocada no arquivo `/etc/hosts`, mais tarde.

7.11. Criando o arquivo /etc/hosts

Se uma placa de rede será configurada, você precisa decidir qual endereço IP, FQDN e apelidos (aliases) a serem utilizados e que devem ser colocados no arquivo `/etc/hosts`. A sintaxe é esta:

```
<endereço IP> meuservidor.examplo.org apelidos
```

Você deve ter certeza de que o endereço IP está na faixa de endereços privados. Faixas válidas são:

```
        Class Networks
        A     10.0.0.0
        B     172.16.0.0 through 172.31.0.255
        C     192.168.0.0 through 192.168.255.255
```

Um endereço IP válido pode ser 192.168.1.1. Um FQDN válido para este IP pode ser www.linuxfromscratch.org. (não recomendado o uso deste endereço porque este é um endereço registrado e válido de domínio e pode causar problemas com o domain name server).

Mesmo que você não vá utilizar uma placa de rede, você ainda precisa fornecer um FQDN. Isto é necessário para certos programas funcionarem corretamente.

Crie o arquivo `/etc/hosts` executando:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
[192.168.1.1] [<HOSTNAME>.example.org] [HOSTNAME]

# End /etc/hosts (network card version)
EOF
```

O *`[192.168.1.1]`* e *`[<HOSTNAME>.example.org]`* precisam ser alterados de acordo com seu gosto (ou requisitos, se for um endereço IP determinado por um administrador de sistemas/redes e se esta máquina for conectada à uma rede existente).

Se uma placa de rede não for configurada, crie o arquivo `/etc/hosts` executando:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (no network card version)

127.0.0.1 [<HOSTNAME>.example.org] [HOSTNAME] localhost

# End /etc/hosts (no network card version)
EOF
```

7.12. Configurando o script de rede

Esta seção somente aplica-se a quem irá configurar uma placa de rede.

Se você não possui nenhuma placa de rede, você provavelmente não irá criar nenhum arquivo de configuração relacionado a placas de rede. Se este é o caso, você deve remover as ligações simbólicas `network` de todos os diretórios de níveis de execução (`/etc/rc.d/rc*.d`).

7.12.1. Criando o arquivo de configuração da interface de rede

Quais interfaces de rede são ativadas e quais são desativadas pelo script network depende dos arquivos no diretório `/etc/sysconfig/network-devices`. Este diretório deve conter arquivos na forma `ifconfig.xyz`, onde “xyz” é um nome de interface de rede (como eth0 ou eth0:1). Dentro deste diretório ficam os arquivos que definem os atributos desta interface, tal como seu endereçamento IP, máscaras de subrede, e assim por diante.

O seguinte comando criará um arquivo `ipv4` de exemplo para o dispositivo *eth0* device:

```
cd /etc/sysconfig/network-devices &&
mkdir ifconfig.eth0 &&
cat > ifconfig.eth0/ipv4 << "EOF"
ONBOOT=yes
SERVICE=ipv4-static
IP=192.168.1.1
GATEWAY=192.168.1.2
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Claro, os valores destas variáveis precisam ser alterados em cada arquivo para refletir a configuração apropriada. Se a variável `ONBOOT` é ajustada para “yes”, o script network irá ativar a Network Interface Card (NIC) correspondente durante a inicialização do sistema. Se ajustado para qualquer valor diferente de “yes”, este será ignorado pelo script e, portanto, não será ativada

A variável `SERVICE` define o método usado para obtenção do endereço IP. O pacote LFS-Bootscripts tem um formato modular de atribuição do IP, e cria arquivos adicionais no diretório `/etc/sysconfig/network-devices/services` permitindo outros métodos de atribuição do IP. Isto é normalmente usado para o Dynamic Host Configuration Protocol (DHCP), que é implementado no livro BLFS.

A variável `GATEWAY` deve conter o endereço IP do gateway padrão, se algum estiver presente. Do contrário, comente (marque com #) a variável.

A variável `PREFIX` contém o número de bits usados na subnet. Cada octeto em um endereço IP são 8 bits. Se a máscara de rede (netmask) da subnet for 255.255.255.0, então se está usando os primeiros três octetos (24 bits) para especificar o número na rede (network number). Se o netmask fosse 255.255.255.240, estaria sendo usados então os primeiros 28 bits. Os prefixos com tamanho de de 24 bits são usados geralmente pelo DSL e pelo Internet Service Providers (ISPs). Neste exemplo (PREFIX=24), a máscara de rede (netmask) é 255.255.255.0. Ajuste a variável `PREFIX` de acordo com seu subnet

7.12.2. Criando o arquivo /etc/resolv.conf

Se o sistema for conectado à Internet, necessitará de algumas configurações para lidar com o Domain Name Service (DNS) para converter nomes de domínio da Internet em endereços IP, e vice versa. O melhor a fazer é colocar o endereço IP servidor DNS, disponível no ISP ou com o administrador da rede, no arquivo `/etc/resolv.conf`. Crie o arquivo executando o seguinte comando:

```
cat > /etc/resolv.conf << "EOF"
# Begin /etc/resolv.conf

domain {[Your Domain Name]}
nameserver [IP address of your primary nameserver]
nameserver [IP address of your secondary nameserver]

# End /etc/resolv.conf
EOF
```

Substitua *`[IP address do the nameserver]`* pelo endereço IP do servidor DNS. Haverá freqüentemente mais de uma entrada (é recomendável servidores secundários para uma eventual indisponibilidade do servidor). Se você necessitar ou quiser apenas um servidor DNS, remova a segunda linha *nameserver* do arquivo. O endereço IP pode também ser um roteador na rede lo.

Chapter 8. Fazendo o sistema LFS inicializável

8.1. Introdução

É hora de fazer o sistema LFS inicializável. Este capítulo discute a criação de um arquivo `fstab`, a configuração de um novo kernel para o sistema LFS e a instalação do boot loader GRUB, de modo que o sistema LFS possa ser selecionado durante a inicialização.

8.2. Criando o arquivo /etc/fstab

O arquivo `/etc/fstab` é usado por alguns programas determinar onde os sistemas de arquivos devem ser montados, em que ordem e como devem ser verificados (em sua integridade) antes da montagem. Crie uma nova tabela dos sistemas de arquivos como esta:

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

# file system  mount-point  type   options         dump  fsck
#                                                        order

/dev/[xxx]     /            [fff]  defaults        1     1
/dev/[yyy]     swap         swap   pri=1           0     0
proc           /proc        proc   defaults        0     0
sysfs          /sys         sysfs  defaults        0     0
devpts         /dev/pts     devpts gid=4,mode=620  0     0
shm            /dev/shm     tmpfs  defaults        0     0
# End /etc/fstab
EOF
```

Substitua *`[xxx]`*, *`[yyy]`* e *`[fff]`* pelos valores apropriados para o seu sistema, por exemplo `hda2`, `hda5` e `ext2`. Para detalhes sobre o conteúdo dos seis campos neste arquivo, veja **man 5 fstab**.

Ao usar um sistema de arquivo com journalling, o *`1 1`* no final da linha respectiva deve ser substituído por *`0 0`* porque esta partição não necessita ser verificada.

O ponto da montagem `/dev/shm` para `tmpfs` é incluído para habilitar o compartilhamento de memória POSIX. O kernel deve ter o necessário suporte configurado nele para que esta funcione (há mais informações sobre isso na seção seguinte). Note por favor que muito poucos softwares atualmente usam a memória compartilhada POSIX. Por isso, considere opcional ponto de montagem `/dev/shm`. Para mais informação, veja `Documentation/filesystems/tmpfs.txt` na árvore de diretório dos fontes do kernel.

Há outras linhas que podem ser adicionadas ao arquivo `/etc/fstab`. Um exemplo é uma linha para dispositivos USB:

```
usbfs        /proc/bus/usb usbfs   devgid=14,devmode=0660 0 0
```

Esta opção somente terá utilidade se o "Support for Host-side USB" e o "USB device filesystem" estiver configurarado no kernel. Se "Support for Host-side USB" estiver compilado como um módulo, então o `usbcore` deve estar listado no arquivo `/etc/sysconfig/modules`.

8.3. Linux-2.6.11.12

O pacote Linux contém o kernel Linux.

Tempo de compilação aproximado: 4.20 SBU
Espaço em disco necessário: 181 MB

Requisitos de instalação: Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Modutils, Perl e Sed

8.3.1. Instalação do kernel

A instalação do kernel envolve alguns passos — configuração, compilação e instalação. Há algumas alternativas para configurar o kernel. Leia o arquivo `README` que acompanha o código-fonte do kernel e veja métodos alternativos à maneira que este livro configura o kernel.

Prepare para a compilação com o seguinte comando:

```
make mrproper
```

Isto garante que o código-fonte esteja limpo para a compilação. Os desenvolvedores do kernel recomendam que este comando seja executado antes de cada compilação do kernel. Nada garante que o kernel esteja limpo após a descompactação.

Se na Section 7.6, "Configurando o terminal Linux," você se decidiu por compilar o keymap no kernel, execute o comando seguinte:

```
loadkeys -m /usr/share/kbd/keymaps/[path to  keymap] > \
    drivers/char/defkeymap.c
```

Por exemplo, se você usando um teclado holandês, use `/usr/share/kbd/keymaps/i386/qwerty/nl.map.gz`.

Configure o kernel através de uma interface orientada por menus. O livro do BLFS tem alguma informação a respeito das exigências particulares quando à configuração do kernel para alguns pacotes que não integram o LFS. Leia em *http://www.linuxfromscratch.org/blfs/view/svn/ longindex.html#kernel-config-index*:

```
make menuconfig
```

Como alternativa, executar o **make oldconfig** pode ser mais apropriado em algumas situações. Veja o arquivo `README` para mais informação.

Se preferir, você pode pular a configuração do kernel, copiando o arquivo, `.config`, do seu sistema anfitrião (assumindo que ele está disponível) para o diretório `linux-2.6.11.12`. Entretanto, nós não recomendamos esta opção. É muito melhor explorar todos os menus de configuração e criar uma configuração do kernel totalmente do zero.

Note

O NPTL exige que o kernel seja compilado com o GCC-3.x, no nosso caso estamos usando a versão 3.4.3. Não se recomenda compilar o kernel com o GCC-2.95.x, porque isto causa falhas no suite do teste do Glibc. Normalmente, isto não seria mencionado porque o LFS não configurou um

GCC-2.95.x. Infelizmente, a documentação do kernel está defasada e ainda se refere ao Gcc-2.95.3 como o compilador recomendado.

Compile a imagem e os módulos do kernel:

```
make
```

Para usar módulos do kernel, o arquivo `/etc/modprobe.conf` pode ser necessário. Informações pertinentes aos módulos e à configuração do kernel podem ser encontradas na documentação do kernel, no diretório `linux-2.6.11.12/Documentation`. A leitura do `modprobe.conf(5)`também pode ser interessante.

Tenha muito cuidado ao ler algum outra documentação sobre os módulos do kernel porque geralmente se refere somente ao kernel da série 2.4.x. Até onde nós sabemos, características da configuração do kernel específicos para o Hotplug e o Udev não estão documentadas. O problema é que o Udev criará um nó de dispositivo somente se Hotplug ou um script criado pelo usuário inserir o módulo correspondente no kernel, e nem todos os módulos são detectáveis pelo Hotplug. Note que uma instrução como esta mostrada abaixo no arquivo `/etc/modprobe.conf` não produz nenhum efeito com o Udev:

```
alias char-major-XXX some-module
```

Por causa das complicações com o uso do Hotplug, Udev, e módulos, nós recomendamos fortemente começar com uma configuração completamente não-modular do kernel, especialmente se esta é a primeira vez que você está usando o Udev.

Instale os módulos, se a configuração do kernel usar módulos:

```
make modules_install
```

Depois que a compilação do kernel está completa, algumas etapas adicionais são necessárias para terminar a instalação. Alguns arquivos precisam ser copiados para o diretório `/boot`.

O caminho para a imagem do kernel pode variar dependendo da plataforma que está sendo usada. O seguinte comando supõe uma arquitetura x86:

```
cp arch/i386/boot/bzImage /boot/lfskernel-2.6.11.12
```

O `System.map` é um arquivo de símbolos. Ele mapeia todas as funções na API, bem como os endereços das estruturas de dados para o kernel em execução. Execute o seguinte comando para instalar o arquivo .map:

```
cp System.map /boot/System.map-2.6.11.12
```

O arquivo de configuração do kernel `.config` produzido pelo **make menuconfig** acima contém todas as seleções feitas na configuração do kernel que foi compilado. É uma idéia boa manter este arquivo para referências futuras:

```
cp .config /boot/config-2.6.11.12
```

É importante notar que os arquivos no diretório das fontes do kernel não são de propriedade do *root*. Sempre que um pacote é descompactado como usuário *root* (como nós fizemos com o chroot), os arquivos têm os IDs de usuário e de grupo de quem quer que fez o empacotamento no computador de origem. Isto geralmente não é um problema para nenhum outro pacote que seja instalado porque a árvore dos fontes é removida após a instalação. Entretanto, é comum reter a árvore dos fontes do Linux por muito tempo. Por causa disto, existe a possibilidade de que o ID de usuário do empacotador seja atribuído a alguém na sua máquina. Essa pessoa teria então o acesso de escrita ao fontes do kernel.

Se a árvore dos fontes do kernel for mantida, execute o **chown -R 0:0** no diretório `linux-2.6.11.12` para fazer com que todos os arquivos sejam de propriedade do root *root*.

Warning

Algumas documentações do kernel recomendam a criação de um symlink em `/usr/src/linux` apontando para o diretório dos fontes do kernel. Isto é válido especificamente para os kernels das séries anteriores à 2.6 e *não deve* ser criado em um sistema LFS, pois pode causar problemas com pacotes que você pode vir a instalar quando o sistema básico LFS estiver completo..

Também os cabeçalhos do kernel que estão no diretório `include` do sistema devem *sempre* ser os mesmos com os quais o Glibc foi compilado, isto é, aqueles do pacote Linux-Libc-Headers e, conseqüentemente, *nunca* devem ser substituídos pelos cabeçalhos do kernel.

8.3.2. Conteúdo do Linux

Arquivos instalados: config-2.6.11.12, lfskernel-2.6.11.12, and System.map-2.6.11.12

Descrição rápida

`config-2.6.11.12`	Contem todas as seleções feitas na configuração do kernel
`lfskernel-2.6.11.12`	O núcleo do sistema Linux. Quando um computador é ligado e inicia um sistema Linux, a primeira parte carregada é o kernel. O kernel detecta e inicializa os componentes de hardware do sistema (portas seriais e paralelas, placas de som e de rede, controladores IDE e SCSI etc.), em seguida faz estes componentes disponíveis como uma árvore de arquivos para os softwares, e faz de uma única CPU uma máquina com multitasking capaz de executar várias requisições dos programas ao mesmo tempo
`System.map-2.6.11.12`	Uma lista dos endereços e dos símbolos; traça os pontos de entrada e os endereços de todas as funções e estruturas de dados do kernel

8.4. Tornando o sistema LFS inicializável

Seu novíssimo sistema LFS está quase completo. Uma das últimas coisas a fazer é assegurar que o sistema possa ser carregado corretamente. As instruções abaixo aplicam-se somente aos computadores da arquitetura IA-32, ou seja os PCs. Informação sobre a inicialização ("boot loading") para outras arquiteturas devem estar disponíveis nos locais usuais onde se encontram recursos e informações específicas destas arquiteturas.

A inicialização, ou carregamento, do sistema pode ser uma área complexa, assim algumas palavras são oportunas por precaução. Familiarize-se com o boot loader atual e com demais sistemas operacionais existentes no disco-rígido inicializável. Certifique-se de que um disco de emergência está pronto para "resgatar" o computador se ele se tornar não-inicializável.

Mais cedo, nós compilamos e instalamos o software de boot loader GRUB, já nos preparando para esta etapa. O procedimento envolve escrever alguns arquivos especiais do GRUB em diretórios específicos do disco rígido. Nós recomendamos criar um disco flexível do carregador do GRUB como apoio. Coloque um disco flexível em branco no drive e execute os seguintes comandos:

```
dd if=/boot/grub/stage1 of=/dev/fd0 bs=512 count=1
dd if=/boot/grub/stage2 of=/dev/fd0 bs=512 seek=1
```

Remova o disquete e guarde-o em algum lugar seguro. Agora, execute o shell **grub**:

```
grub
```

O GRUB usa sua própria estrutura de nomes para discos e partições no format *(hdn,m)*, onde *n* é o número do disco rígido e o *m* o número da partição, e ambos partem de zero. Por exemplo, a partição `hda1` é *(hd0,0)* para o GRUB e a `hdb3` é *(hd1,2)*. Ao contrário do Linux, o GRUB não considera os drives de CDRom como sendo discos rígidos. Por exemplo, se existir um drive de CD em `hdb` e um segundo disco rígido em `hdc`, para o GRUB este segundo disco rígido seria o *(hd1)*.

Com atenção a estas informações, determine o nome apropriado para a partição root (ou a partição de boot, se uma diferente for usada). No exemplo que segue, supõe-se que a partição root (inicializável) é `hda4`.

Diga ao GRUB onde procurar por seus arquivos `stage{1,2}`. A chave Tab pode ser usada a qualquer tempo para fazer o GRUB mostrar as alternativas:

```
root (hd0,3)
```

Warning

O próximo comando sobrescreve o boot loader atual. Não execute o comando se isto não for desejado, por exemplo, se usar um gerenciador de boot de terceiros para gerenciar a Master Boot Record (MBR). Neste caso, faz mais sentido instalar o GRUB no "boot sector" da partição LFS. Neste caso, o comando seguinte ficaria da seguinte forma **`setup (hd0,3)`**.

Diga o GRUB para instalar-se no MBR do `hda`:

```
setup (hd0)
```

Se todos correr bem, o GRUB foi informado para encontrar seu arquivos no diretório `/boot/grub`. Isto é tudo a se fazer aqui. Saia do shell **grub**:

```
quit
```

Crie um arquivo de “menu” definindo as opções de inicialização do GRUB:

```
cat > /boot/grub/menu.lst << "EOF"
# Begin /boot/grub/menu.lst

# By default boot the first menu entry.
default 0

# Allow 30 seconds before booting the default.
timeout 30

# Use prettier colors.
color green/black light-green/black

# The first entry is for LFS.
title LFS 6.1
root (hd0,3)
kernel /boot/lfskernel-2.6.11.12 root=/dev/hda4
EOF
```

Acrescente uma entrada para a distribuição anfitriã se quiser. Pode se parecer com isto:

```
cat >> /boot/grub/menu.lst << "EOF"
title Red Hat
root (hd0,2)
kernel /boot/kernel-2.6.5 root=/dev/hda3
initrd /boot/initrd-2.6.5
EOF
```

Se mantiver dual-booting com o Windows, a seguinte entrada dará suporte a ele:

```
cat >> /boot/grub/menu.lst << "EOF"
title Windows
rootnoverify (hd0,0)
chainloader +1
EOF
```

Se o **info grub** não contiver informações suficientes, informação adicional a respeito do GRUB pode ser encontrada em: *http://www.gnu.org/software/grub/*.

O FHS prevê que o arquivo `menu.lst` do GRUB deve ter um symlink para `/etc/grub/menu.lst`.Para satisfazer esta exigência, use o seguinte comando:

```
mkdir /etc/grub &&
ln -s /boot/grub/menu.lst /etc/grub
```

Chapter 9. Fim

9.1. Fim

Muito bem! O novo sistema LFS está instalado! Nós desejamos muito sucesso com seu novo sistema Linux personalizado.

Pode ser uma boa idéia criar o arquivo `/etc/lfs-release`. Tendo este arquivo, é muito fácil para você (e para nós se você necessitar pedir ajuda em algum momento) saber que versão do LFS está instalada no seu sistema. Crie este arquivo com o comando:

```
echo 6.1 > /etc/lfs-release
```

9.2. Receba seu número

Agora que você terminou o livro, você quer receber o seu número como um usuário LFS? Visite a página *http://www.linuxfromscratch.org/cgi-bin/lfscounter.cgi* e faça seu registro como um usuário LFS, informando apenas seu nome e a primeira versão do LFS que você usou.

Vamos reiniciar pelo sistema LFS agora.

9.3. Reinicialize o sistema

Agora que todo o software está instalado, é hora de reinicializar seu computador. Entretanto, você deve estar ciente de algumas coisas. O sistema que você criou neste livro é realmente mínimo, e é muito provável que não terá a funcionalidade que necessita para continuar em frente. Instalando alguns pacotes extras do livro BLFS quando ainda em nosso ambiente chroot atual, você pode ficar em uma posição muito melhor para continuar. Instalando um web browser em modo texto, como o Lynx, você pode facilmente ver o livro do BLFS em um terminal virtual, e construir pacotes em outro. O pacote do GPM permitirá também que você execute ações de copia-e-cola em seus terminais virtuais. Por último, se você estiver em uma situação onde a configuração estática do IP não satisfaz suas exigências de networking, instalar pacotes tais como Dhcpcd or PPP neste momento pode também ser muito útil.

Agora que nós dissemos tudo, vamos carregar nossa nossa instalação do LFS pela primeira vez! Primeiro, saia do ambiente chroot:

```
logout
```

Desmonte os sistemas de arquivos virtuais:

```
umount $LFS/dev/pts
umount $LFS/dev/shm
umount $LFS/dev
umount $LFS/proc
umount $LFS/sys
```

Desmonte o sistema de arquivos do LFS:

```
umount $LFS
```

Se partições múltiplas foram criadas, desmonte as outras partições antes de desmontar a principal, assim:

```
umount $LFS/usr
umount $LFS/home
umount $LFS
```

Agora, reinicialize o sistema com:

```
shutdown -r now
```

Supondo que o GRUB foi ajustado como esboçado acima, o menu estará ajustado para *LFS 6.1* automaticamente.

Quando o reboot estiver completo, o sistema LFS está pronto para uso e mais programas podem ser adicionados para servir às suas necessidades.

9.4. E agora?

Obrigado por ler este livro do LFS. Nós esperamos que você ache este livro útil e aprenda mais sobre o processo de criação do sistema.

Agora que o sistema LFS está instalado, você pode querer saber “O que mais?” Para responder a esta pergunta, nós elaboramos uma lista de recursos para você.

- Manutenção

  Os erros constatados e as notas de segurança são relatadas regularmente para todos os softwares instalados. Como o sistema LFS é inteiramente compilado da fonte, é bom você se manter informado de tais relatórios. Há diversos meios on line que enviam estes relatórios, e alguns são os seguintes:

  - Freshmeat.net (*http://freshmeat.net/*)

    O Freshmeat pode notificá-lo (através do email) de versões novas dos pacotes instalados em seu sistema.

  - *CERT* (Computer Emergency Response Team)

    O CERT tem uma lista de discussão que publica os alertas da segurança a respeito dos vários sistemas operacionais e aplicações. As informações de subscrição estão disponíveis em *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    O Bugtraq é uma lista de discussão sobre segurança de computador. Publica descobertas recentes sobre problemas de segurança, e algumas vezes reparos potenciais para eles. A informação de subscrição está disponível em *http://www.securityfocus.com/archive*.

- Beyond Linux From Scratch

  O livro Beyond Linux From Scratch O livro do Beyond Linux From Scratch traz os procedimentos de instalação de diversos programas e softwares além dos limites do LFS. O projeto de BLFS fica situado em *http://www.linuxfromscratch.org/blfs/*.

- LFS Hints

  O LFS Hints é uma coleção dos originais informativos submetidos por voluntários da comunidade LFS. As sugestões estão disponíveis em *http://www.linuxfromscratch.org/hints/list.html*.

- Mailing lists

  Há diversas listas de discussão LFS que você pode subscrever se você tiver necessidade da ajuda, quiser permanecer atualizado com os desenvolvimentos atuais, quiser contribuir com o projeto, e a mais. Veja o Chapter 1 - Mailing Lists para mais informações.

- The Linux Documentation Project

  O objetivo do projeto de documentação do Linux (TLDP) é consolidar toda a documentação do Linux. O TLDP tem uma grande coleção de HOWTOs, de guias, e de páginas man. Fica situado em *http://www.tldp.org/*.

Part IV. Apêndices

Appendix A. Termos e Anacronismos

ABI Interface binária de aplicativo (do inglês, Application Binary Interface)

ALFS Automated Linux From Scratch

ALSA Advanced Linux Sound Architecture

API Interface de Programação de Aplicativos (do inglês, Application Programming Interface)

ASCII American Standard Code for Information Interchange

BIOS Basic Input/Output System

BLFS Beyond Linux From Scratch

BSD Berkeley Software Distribution

chroot change root

CMOS Complementary Metal Oxide Semiconductor

COS Class Of Service

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CVS Concurrent Versions System

DHCP Dynamic Host Configuration Protocol

DNS Domain Name Service

EGA Enhanced Graphics Adapter

ELF Executable and Linkable Format

EOF End of File

EQN equation

EVMS Enterprise Volume Management System

ext2 second extended file system

FAQ Frequently Asked Questions

FHS Filesystem Hierarchy Standard

FIFO First-In, First Out

FQDN Fully Qualified Domain Name

FTP File Transfer Protocol

GB Gibabytes

GCC GNU Compiler Collection

GID Group Identifier

GMT Greenwich Mean Time

GPG GNU Privacy Guard

HTML Hypertext Markup Language

IDE Integrated Drive Electronics

IEEE Institute of Electrical and Electronic Engineers

IO Input/Output

IP Internet Protocol

IPC Inter-Process Communication

IRC Internet Relay Chat

ISO International Organization for Standardization

ISP Internet Service Provider

KB Kilobytes

LED Light Emitting Diode

LFS Linux From Scratch

LSB Linux Standards Base

MB Megabytes

MBR Master Boot Record

MD5 Message Digest 5

NIC Network Interface Card

NLS Native Language Support

NNTP Network News Transport Protocol

NPTL Native POSIX Threading Library

OSS Open Sound System

PCH Cabeçalhos pré-compilados (do inglês, Pre-Compiled Headers)

PCRE Perl Compatible Regular Expression

PID Identificador de processo (do inglês, Process Identifier)

PLFS Pure Linux From Scratch

PTY pseudo terminal

QA Quality Assurance

QOS Qualidade de serviços (do inglês, Quality Of Service)

RAM Memória de acesso aleatório (do inglês, Random Access Memory)

RPC Chamada a procedimento remoto (do inglês, Remote Procedure Call)

RTC Relógio de tempo real (do inglês, Real Time Clock)

SBU Unidade de compilação padrão (do inglês, Standard Build Unit)

SCO The Santa Cruz Operation

SGR Select Graphic Rendition

SHA1 Secure-Hash Algorithm 1

SMP Symmetric Multi-Processor

TLDP Projeto de Documentação do Linux (do inglês, The Linux Documentation Project)

TFTP Protocolo de transferência de arquivos triviais (do inglês, Trivial File Transfer Protocol)

TLS Thread-Local Storage

UID identificador de usuário (do inglês, User Identifier)

umask mascara de criação de arquivo (do inglês, user file-creation mask)

USB Barramento serial universal (do inglês, Universal Serial Bus)

UTC Horário unversal por coordenadas (do inglês, Coordinated Universal Time)

UUID Identificador universal único (do inglês, Universally Unique Identifier)

VC console virtual (do inglês, Virtual Console)

VGA Conjunto de gráficos de vídeo (do inglês, Video Graphics Array)

VT Terminal virtual (do inglês, Virtual Terminal)

Appendix B. Agradecimentos

Nós gostaríamos de agradecer as pessoas e organizações a seguir pelo contribuição ao projeto Linux From Scratch.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – Criador do LFS, Líder de Projeto do LFS
- *Matthew Burgess* <matthew@linuxfromscratch.org> – LFS Project Leader, LFS Technical Writer/Editor, LFS Release Manager
- *Archaic* <archaic@linuxfromscratch.org> – Escritor e Editor Técnico do LFS, Líder de Projeto do HLFS, Editor do BLFS, Mantedor do Projeto Hints and Patches
- *Nathan Coulson* <nathan@linuxfromscratch.org> – Mantedor do LFS-Bootscripts
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – Líder de Projeto do BLFS
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – Mantedor do LFS, BLFS, HLFS XML e XSL
- *Jim Gifford* <jim@linuxfromscratch.org> – Escritor Técnico do LFS, Líder do Projeto Patches
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – Escritor Técnico do LFS, Mantedor do LFS LiveCD, Líder de Projeto do ALFS
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Mantedor dos Backend-Scripts do Website
- *Ryan Oliver* <ryan@linuxfromscratch.org> – Mantedor do LFS Toolchain
- *James Robertson* <jwrober@linuxfromscratch.org> – Mantedor do Bugzilla
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – Editor do Livro BLFS e Líder de Projeto do Hints and Patches
- Incontáveis outras pessoas nas várias listas LFS e BLFS que ajudaram a fazer esse livro possível dando sugestões, testando o livro, reportando erros, enviando instruções e experiências de instalação dos vários pacotes.

Tradutores

- *Manuel Canales Esparcia* <macana@lfs-es.com> – Projeto de tradução do LFS para Espanhol
- *Johan Lenglet* <johan@linuxfromscratch.org> – Projeto de tradução do LFS para o Francês
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Projeto de tradução do LFS para o Português
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – Projeto de tradução do LFS para o Alemão

Mantedores de Espelhos

Espelhos na América do Norte

- *Scott Kveton* <scott@osuosl.org> – lfs.oregonstate.edu

- *Mikhail Pastukhov* <miha@xuy.biz> – lfs.130th.net
- *William Astle* <lost@l-w.net> – ca.linuxfromscratch.org
- *Jeremy Polen* <jpolen@rackspace.com> – us2.linuxfromscratch.org
- *Tim Jackson* <tim@idge.net> – linuxfromscratch.idge.net
- *Jeremy Utley* <jeremy@linux-phreak.net> – lfs.linux-phreak.net

Espelhos na América do Sul

- *Andres Meggiotto* <sysop@mesi.com.ar> – lfs.mesi.com.ar
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – lfsmirror.lfs-es.info
- *Eduardo B. Fonseca* <ebf@aedsolucoes.com.br> – br.linuxfromscratch.org

Espelhos na Europa

- *Barna Koczka* <barna@siker.hu> – hu.linuxfromscratch.org
- *UK Mirror Service* – linuxfromscratch.mirror.ac.uk
- *Martin Voss* <Martin.Voss@ada.de> – lfs.linux-matrix.net
- *Guido Passet* <guido@primerelay.net> – nl.linuxfromscratch.org
- *Bastiaan Jacques* <baafie@planet.nl> – lfs.pagefault.net
- *Roel Neefs* <lfs-mirror@linuxfromscratch.rave.org> – linuxfromscratch.rave.org
- *Justin Knierim* <justin@jrknierim.de> – www.lfs-matrix.de
- *Stephan Brendel* <stevie@stevie20.de> – lfs.netservice-neuss.de
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – at.linuxfromscratch.org
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – se.linuxfromscratch.org
- *Parisian sysadmins* <archive@doc.cs.univ-paris8.fr> – www2.fr.linuxfromscratch.org
- *Alexander Velin* <velin@zadnik.org> – bg.linuxfromscratch.org
- *Dirk Webster* <dirk@securewebservices.co.uk> – lfs.securewebservices.co.uk
- *Thomas Skyt* <thomas@sofagang.dk> – dk.linuxfromscratch.org
- *Simon Nicoll* <sime@dot-sime.com> – uk.linuxfromscratch.org

Espelhos na Ásia

- *Pui Yong* <pyng@spam.averse.net> – sg.linuxfromscratch.org
- *Stuart Harris* <stuart@althalus.me.uk> – lfs.mirror.intermedia.com.sg

Espelhos na Austrália

- *Jason Andrade* <jason@dstc.edu.au> – au.linuxfromscratch.org

Ex-membros do Time de Projeto

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – Editor do Livro LFS
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Desenvolvedor do Website, e Mantedor do FAQ
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Mantedor do Wiki
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – Mantedor do Servidor NNTP do LFS
- *Alexander Patrakov* <semzx@newmail.ru> – Escritor Técnico do LFS
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – Escritor Técnico do LFS, Mantedor do Bugzilla, Mantedor do LFS-Bootscripts
- *Zack Winkles* <zwinkles@gmail.com> – Ex-escritor Técnico do LFS

Um obrigado muito especial para as pessoas que doam

- *Dean Benson* <dean@vipersoft.co.uk> por várias contribuições comentários
- *Hagen Herrschaft* <hrx@hrxnet.de> por doar um sistema P4 de 2.2 GHz, agora rodando sob o nome Lorien
- *VA Software* que, através do *Linux.com*, doou uma estação de trabalho VA Linux 420 (conhecida como StartX SP2)
- Mark Stone pela doação o Belgarath, o servidor do linuxfromscratch.org

Index

Packages

Programs

Libraries

Scripts

Others